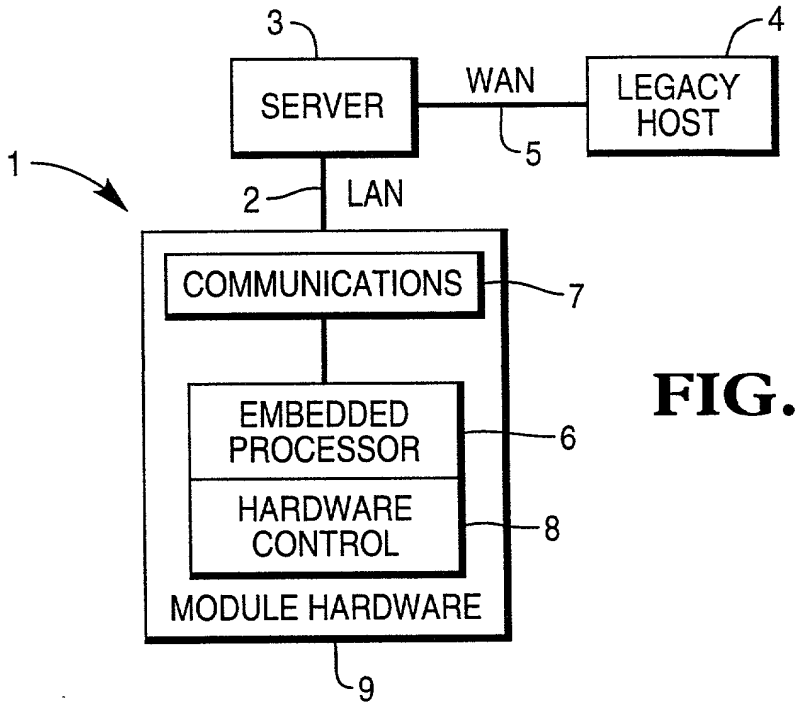

3
SERVER
WAN
4
LEGACY HOST
5
1
2
LAN
COMMUNICATIONS
7
EMBEDDED PROCESSOR
6
HARDWARE CONTROL
8
MODULE HARDWARE
9

FIG. 1

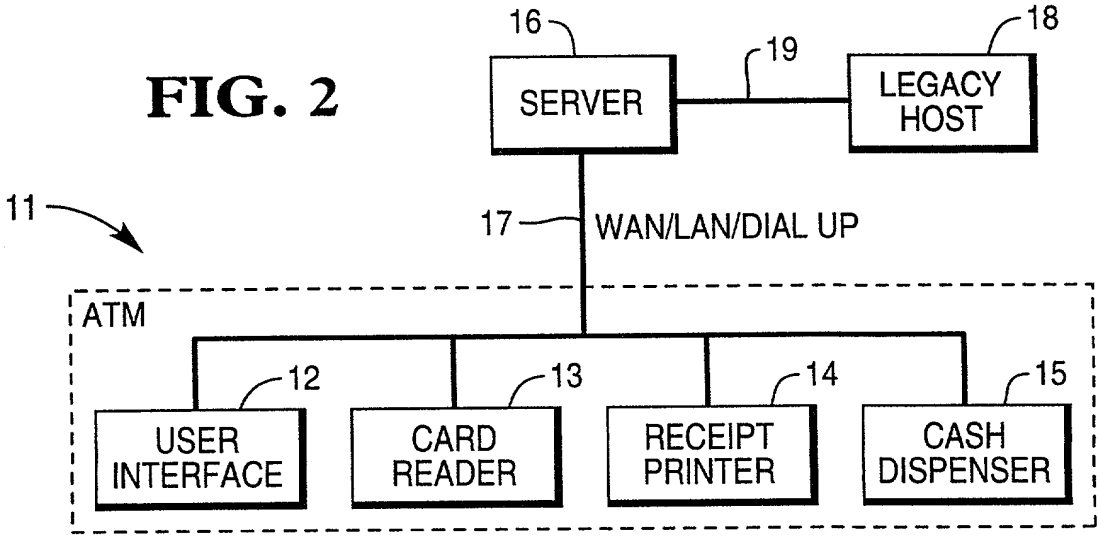

16
SERVER
19
18
LEGACY HOST
11
17
WAN/LAN/DIAL UP
ATM
12
USER INTERFACE
13
CARD READER
14
RECEIPT PRINTER
15
CASH DISPENSER

FIG. 2

FIG. 3

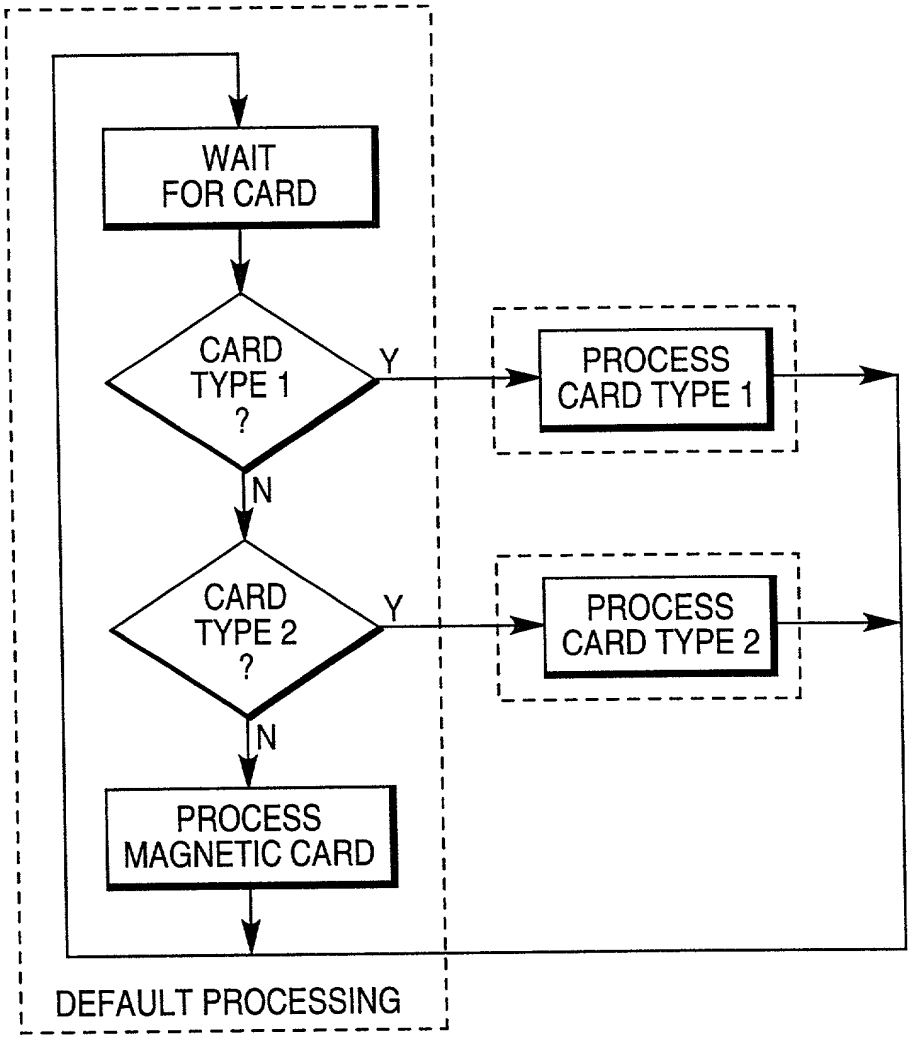

WAIT
FOR CARD
CARD
TYPE 1
?
Y
N
PROCESS
CARD TYPE 1
CARD
TYPE 2
?
Y
N
PROCESS
CARD TYPE 2
PROCESS
MAGNETIC CARD
DEFAULT PROCESSING

FIG. 4a

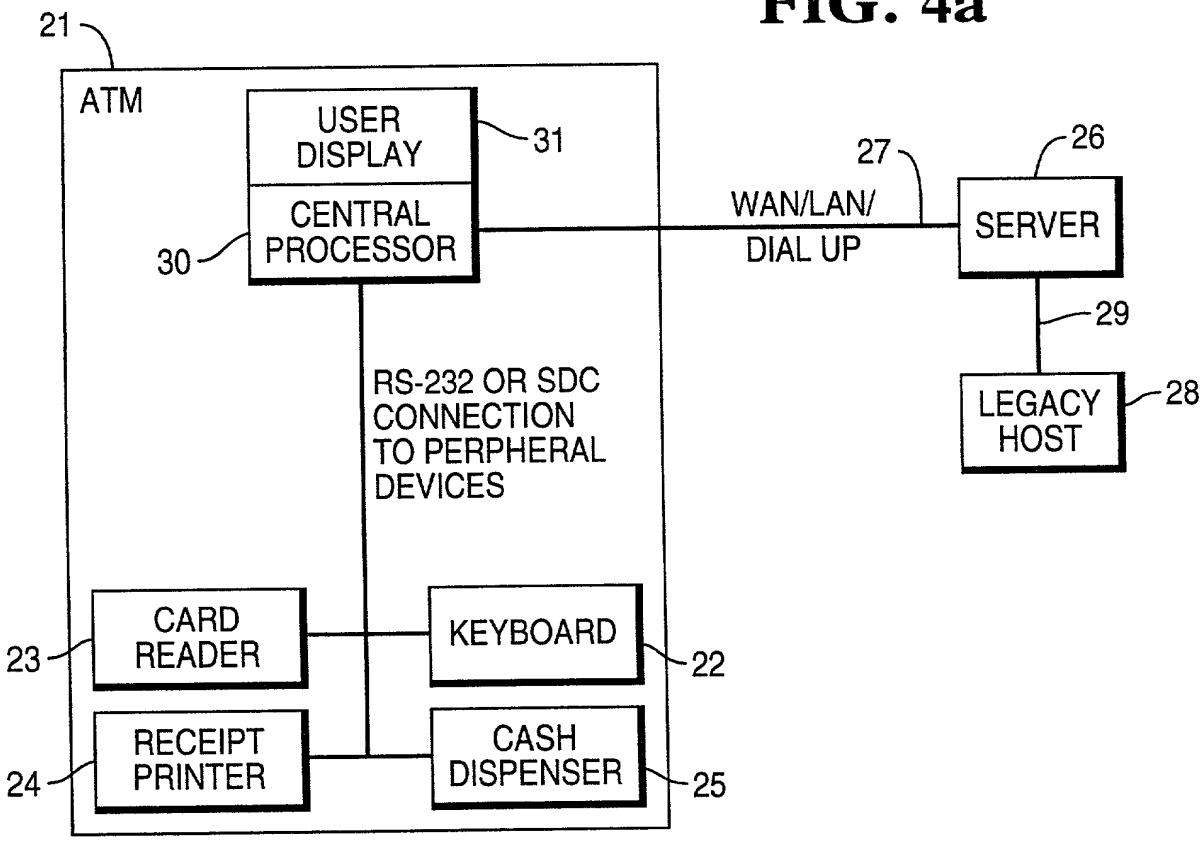


FIG. 4b

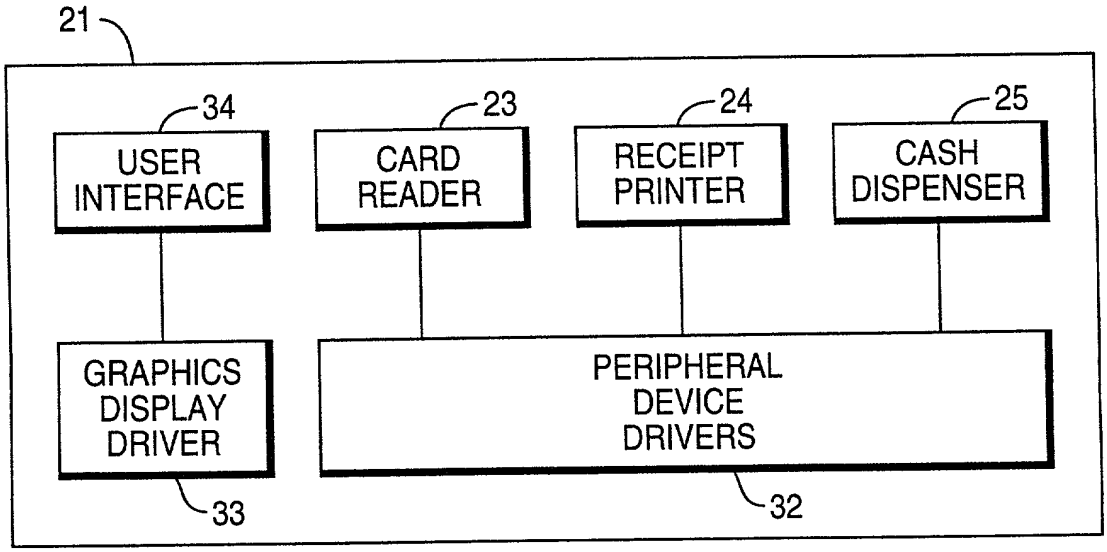


FIG. 5

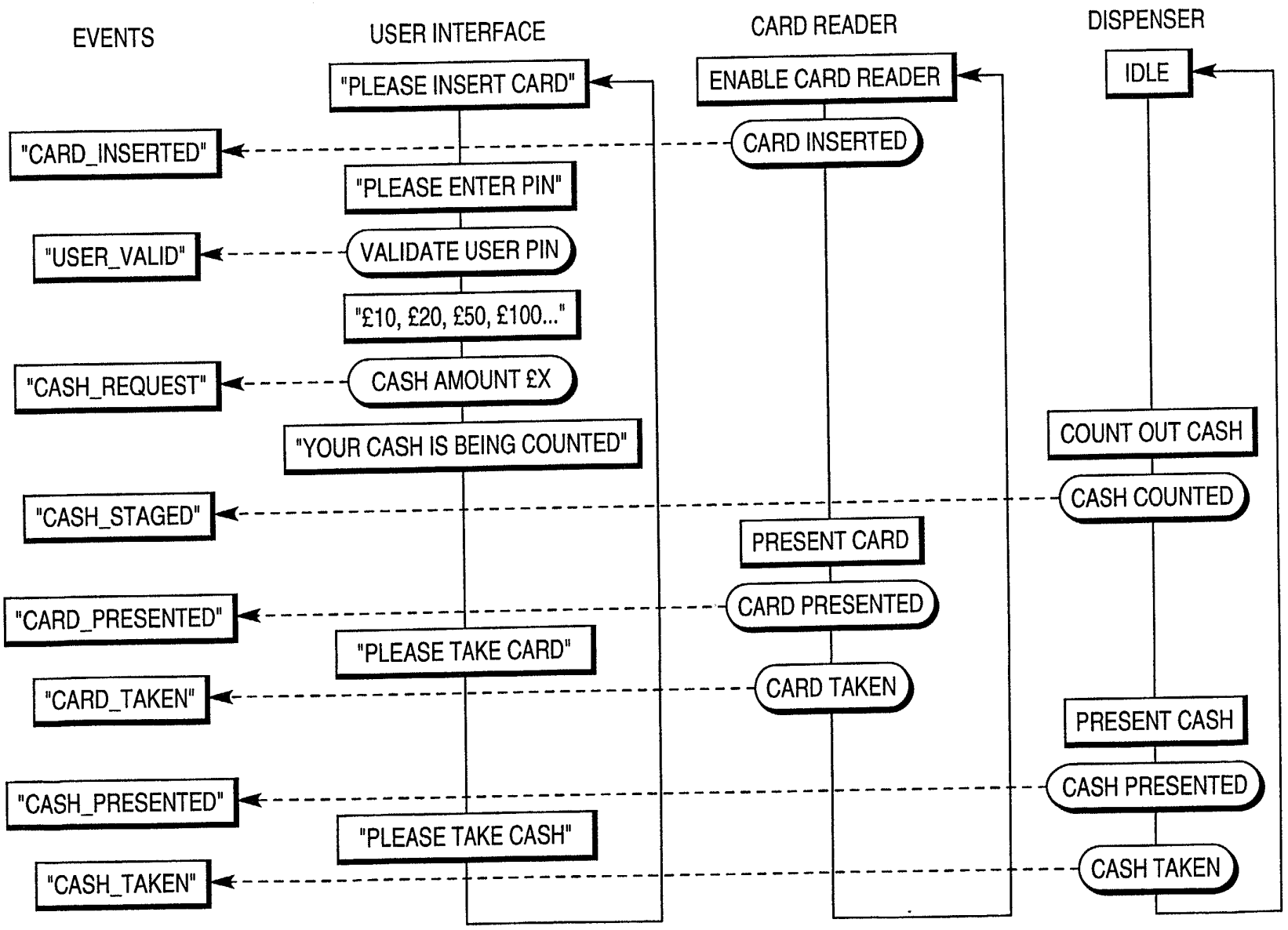
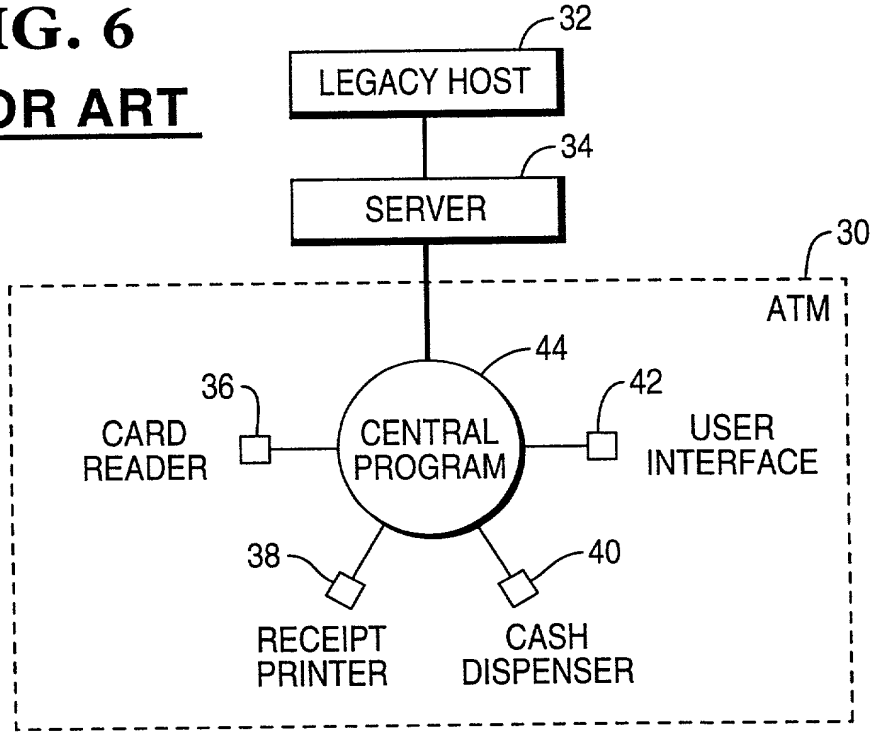


FIG. 6
PRIOR ART



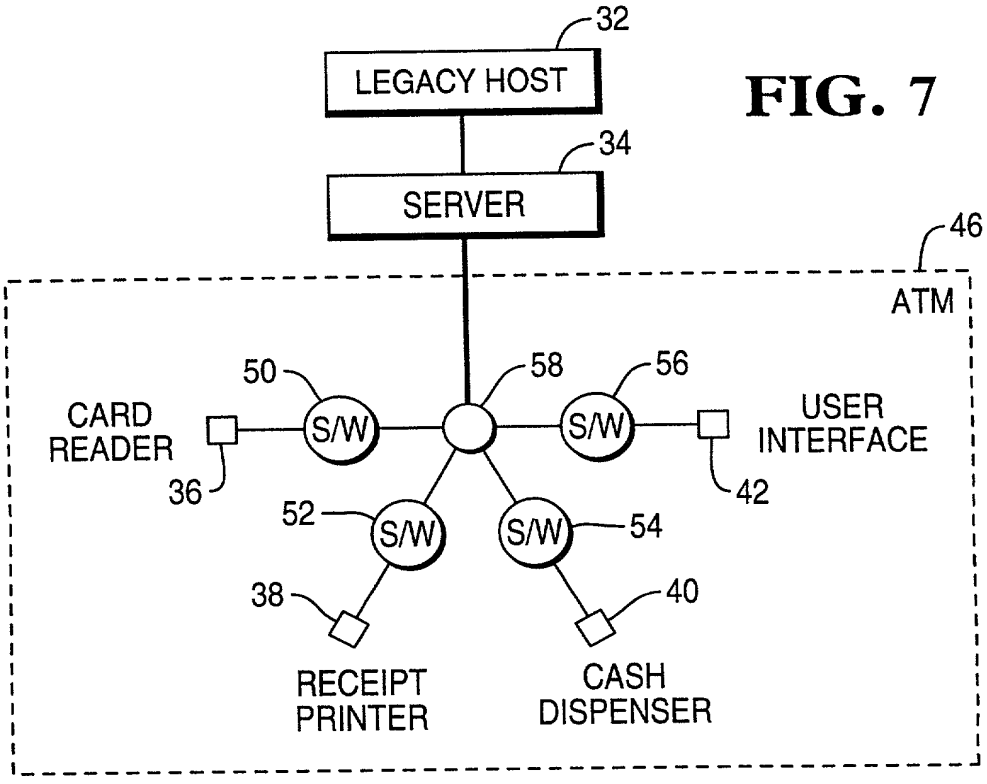


FIG. 7

FIG. 8

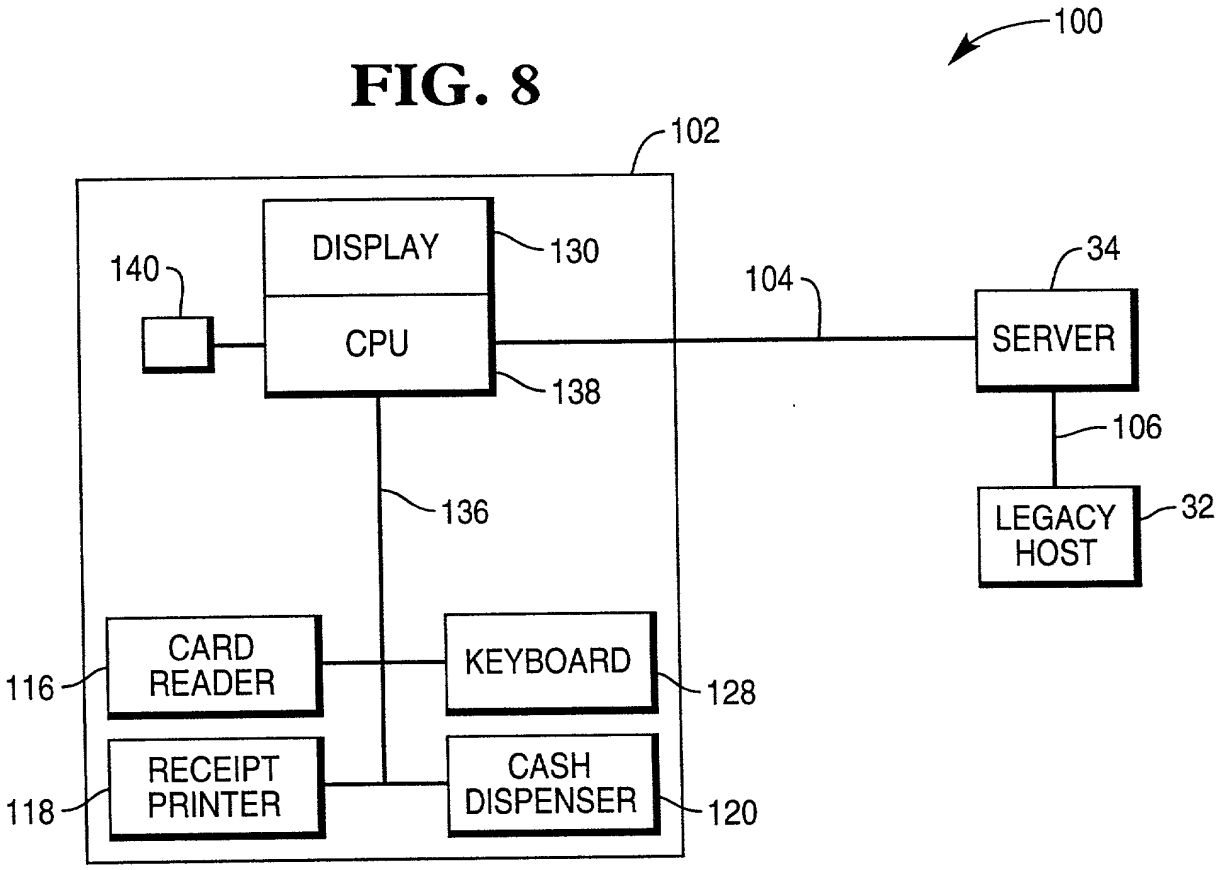


FIG. 11

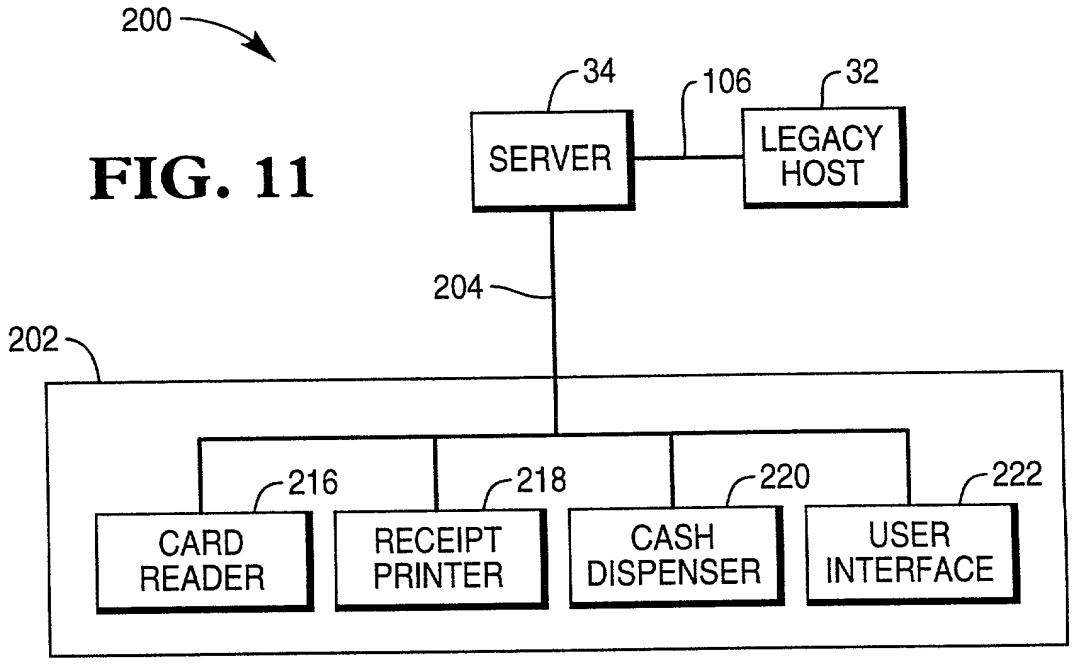


FIG. 9A

150

PERIPHERAL DEVICE	PROCESSOR ADDRESS	PORT
RECEIPT PRINTER	178.132.152.212	6000
CASH DISPENSER	178.132.152.212	6010
KEYBOARD	178.132.152.212	6020
DISPLAY	178.132.152.212	6030
SELF (CARD READER)	178.132.152.212	6040

152 154 156

FIG. 9B

150

152 PERIPHERAL DEVICE	154 PROCESSOR ADDRESS	156 PORT
SELF (CARD READER)	178.132.152.212	6040

FIG. 9C

150

PERIPHERAL DEVICE	PROCESSOR ADDRESS	PORT
CASH DISPENSER	178.132.152.212	6010
SELF (CARD READER)	178.132.152.212	6040

152 154 156

FIG. 12

150'

152 PERIPHERAL DEVICE	154 PROCESSOR ADDRESS	156 PORT
RECEIPT PRINTER	178.132.152.38	6000
CASH DISPENSER	178.132.152.42	6010
KEYBOARD	178.132.152.43	6020
DISPLAY	178.132.152.47	6030
SELF (CARD READER)	178.132.152.52	6040

FIG. 10

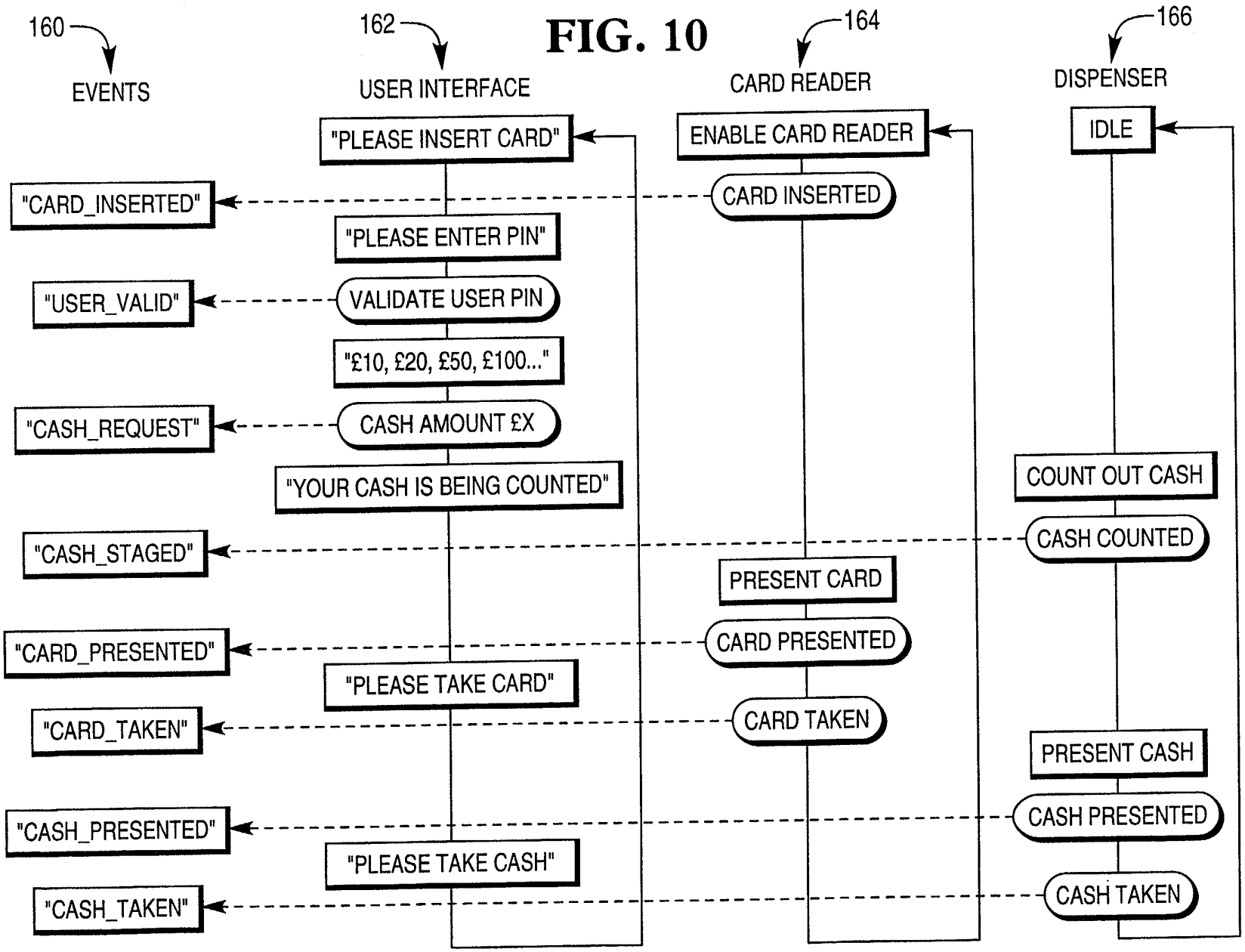


FIG. 13

PRIOR ART

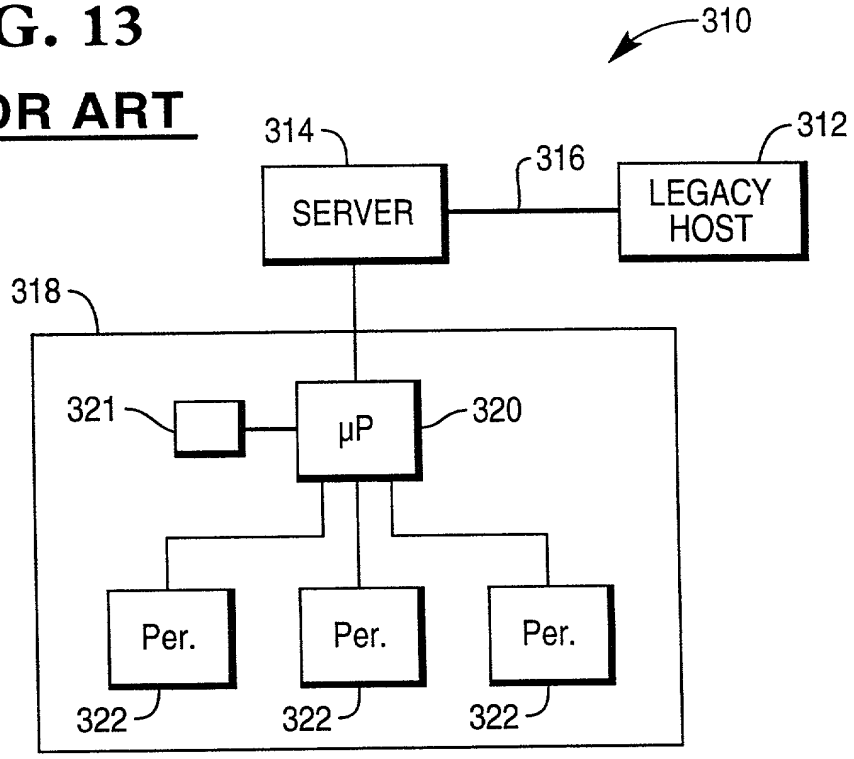


FIG. 14

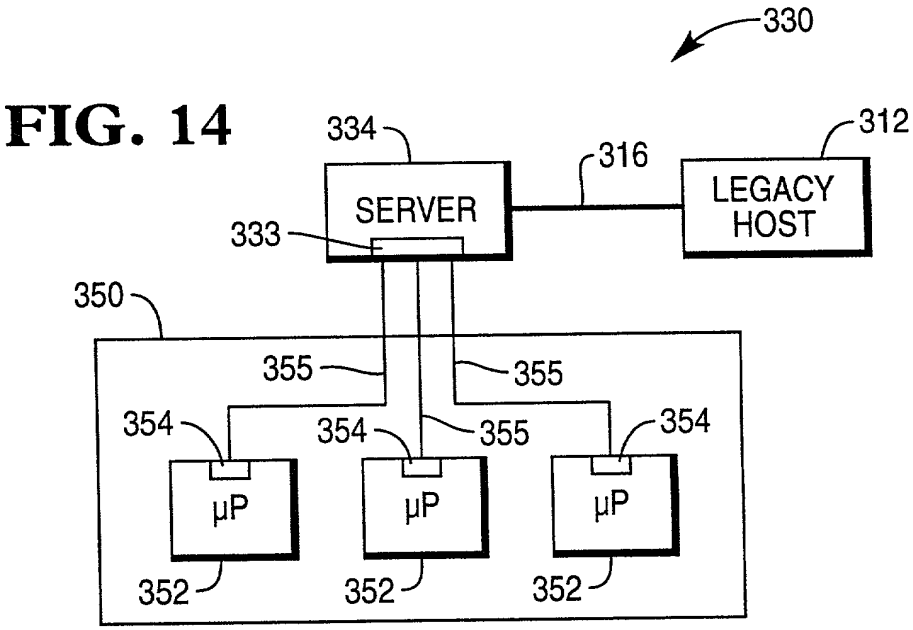


FIG. 15

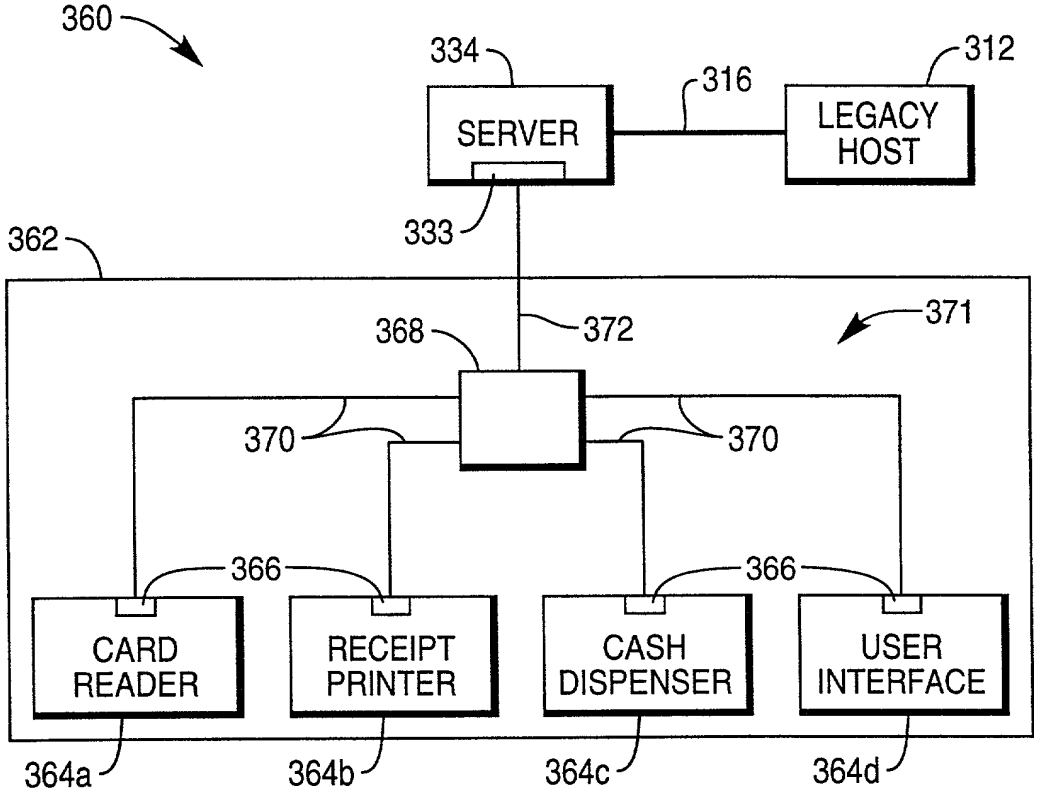
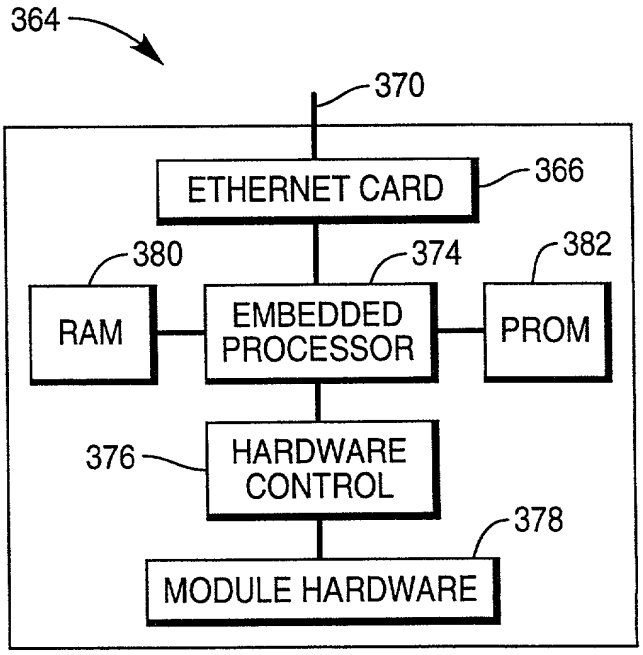

360
334
312
316
SERVER
LEGACY HOST
333
362
371
368
372
370
370
366
366
CARD READER
RECEIPT PRINTER
CASH DISPENSER
USER INTERFACE
364a
364b
364c
364d

FIG. 16


364
370
ETHERNET CARD
366
380
374
382
RAM
EMBEDDED PROCESSOR
PROM
376
HARDWARE CONTROL
378
MODULE HARDWARE

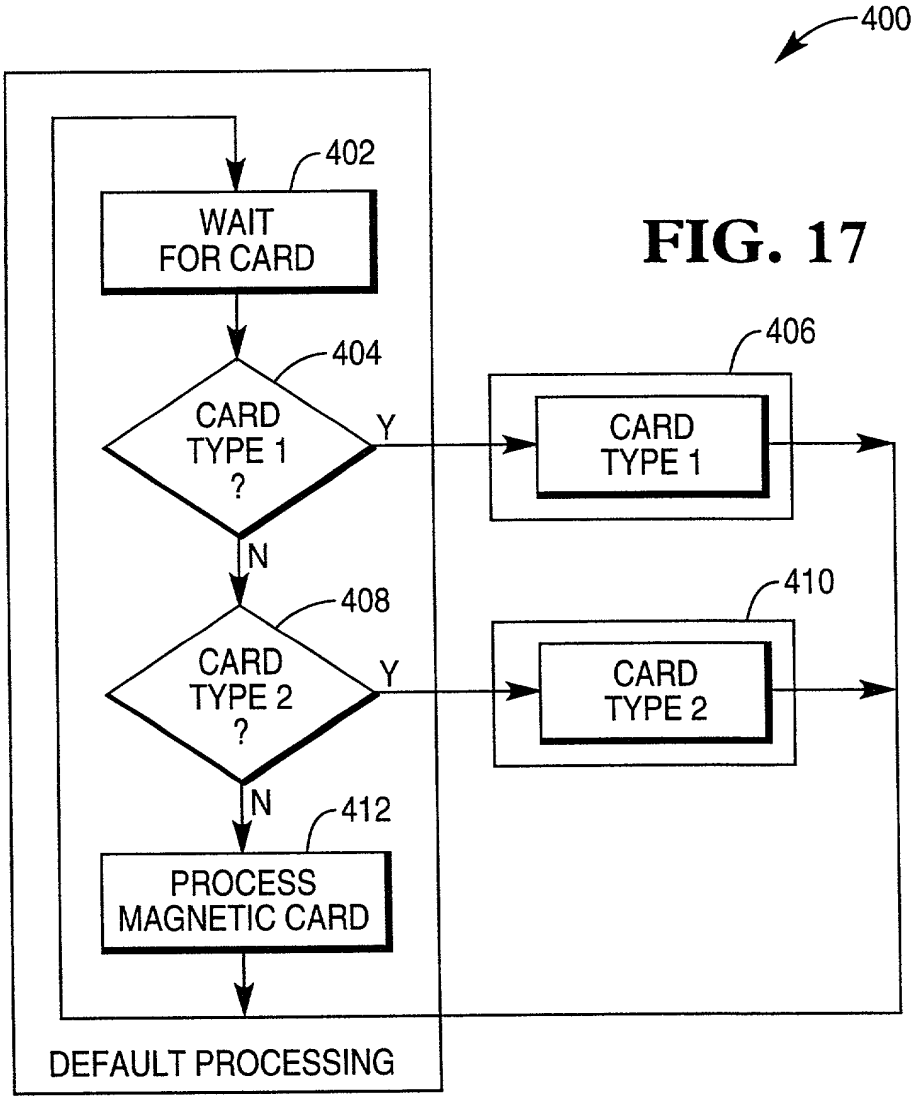

400
402
WAIT FOR CARD
404
CARD TYPE 1 ?
Y
406
CARD TYPE 1
N
408
CARD TYPE 2 ?
Y
410
CARD TYPE 2
N
412
PROCESS MAGNETIC CARD
DEFAULT PROCESSING

FIG. 17

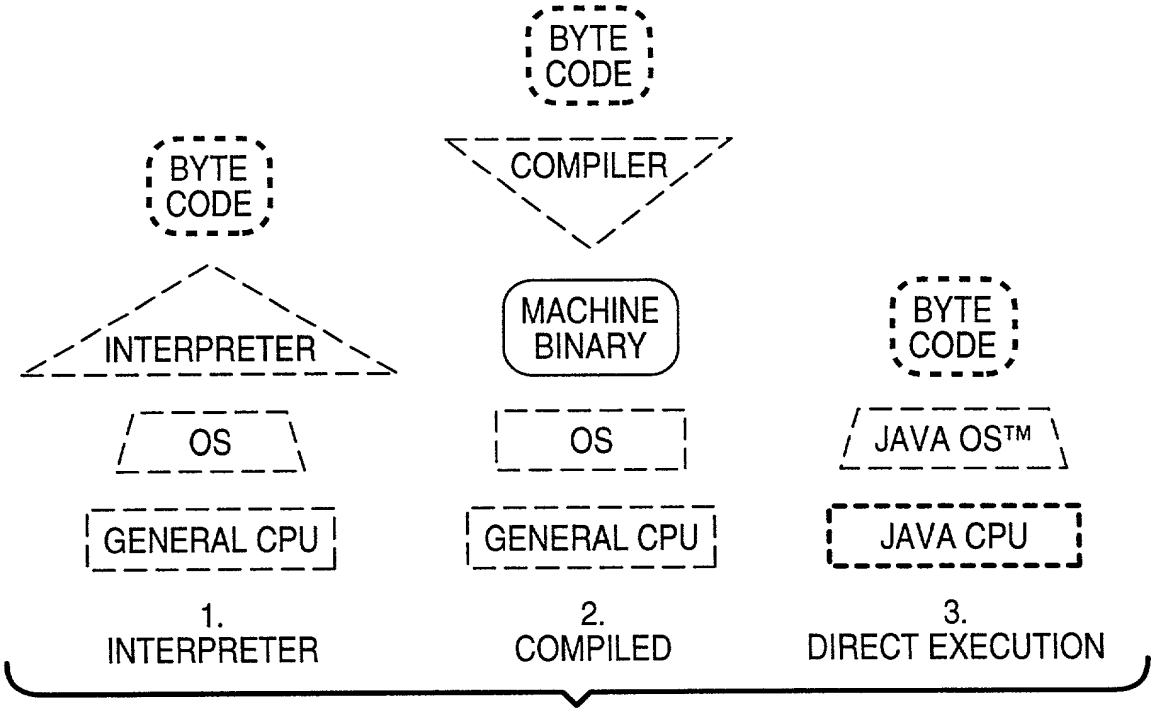

BYTE CODE
BYTE CODE
COMPILER
INTERPRETER
MACHINE BINARY
BYTE CODE
OS
OS
JAVA OS™
GENERAL CPU
GENERAL CPU
JAVA CPU
1. INTERPRETER
2. COMPILED
3. DIRECT EXECUTION

FIG. 18

FIG. 19

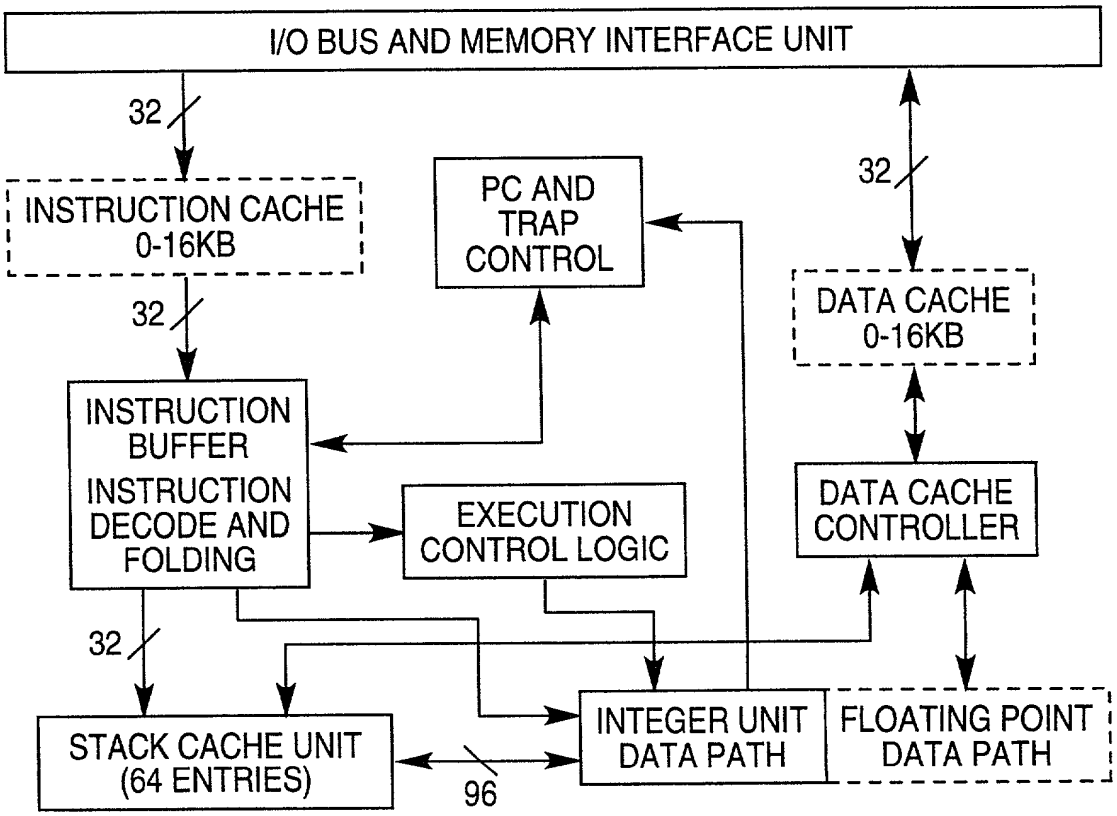

I/O BUS AND MEMORY INTERFACE UNIT
32
32
INSTRUCTION CACHE 0-16KB
PC AND TRAP CONTROL
DATA CACHE 0-16KB
32
INSTRUCTION BUFFER
INSTRUCTION DECODE AND FOLDING
EXECUTION CONTROL LOGIC
DATA CACHE CONTROLLER
32
STACK CACHE UNIT (64 ENTRIES)
96
INTEGER UNIT DATA PATH
FLOATING POINT DATA PATH

FIG. 20

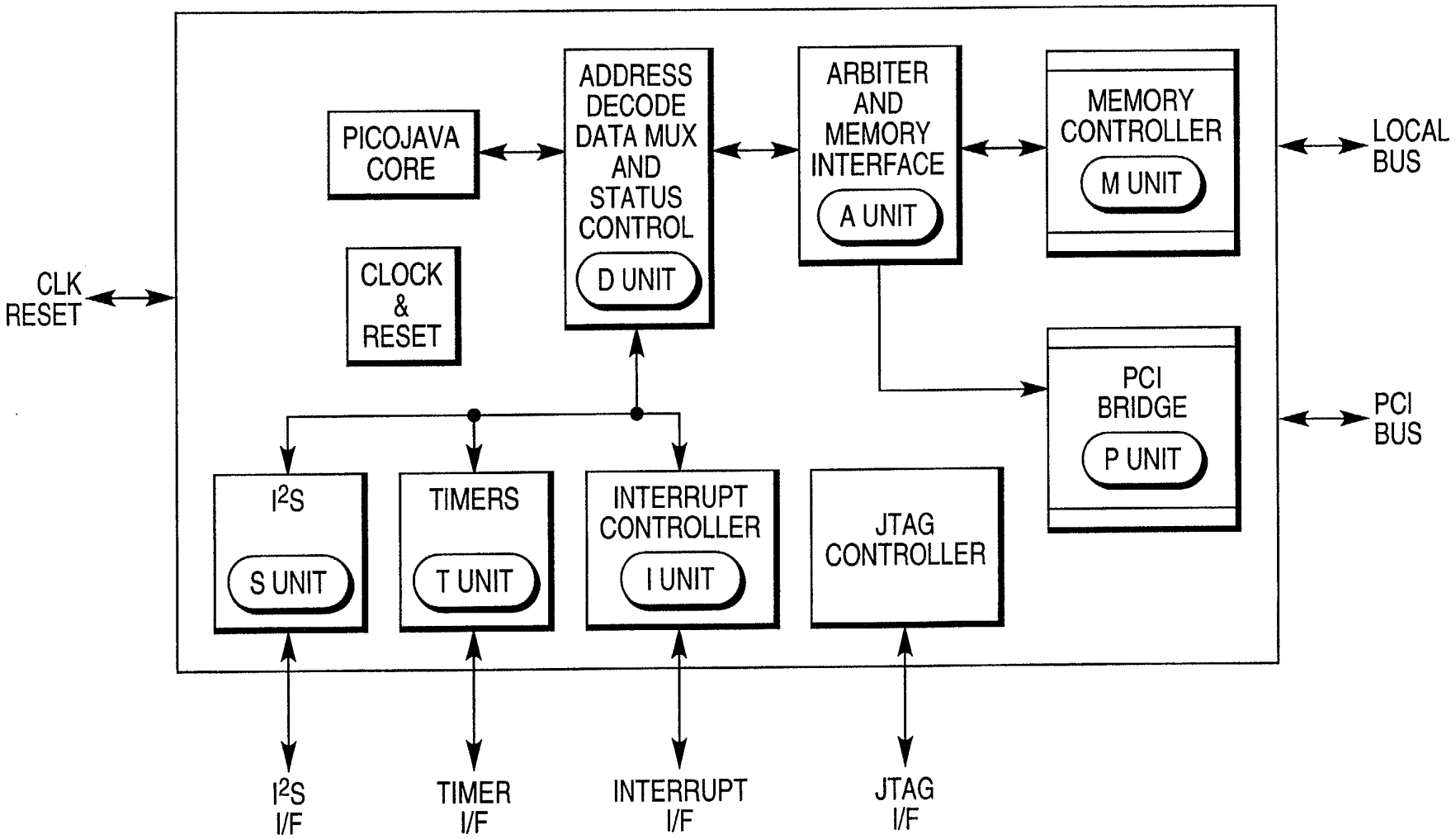

CLK
RESET
PICOJAVA CORE
CLOCK & RESET
ADDRESS DECODE DATA MUX AND STATUS CONTROL
D UNIT
ARBITER AND MEMORY INTERFACE
A UNIT
MEMORY CONTROLLER
M UNIT
LOCAL BUS
PCI BRIDGE
P UNIT
PCI BUS
I2S
S UNIT
TIMERS
T UNIT
INTERRUPT CONTROLLER
I UNIT
JTAG CONTROLLER
I2S I/F
TIMER I/F
INTERRUPT I/F
JTAG I/F

FIG. 21

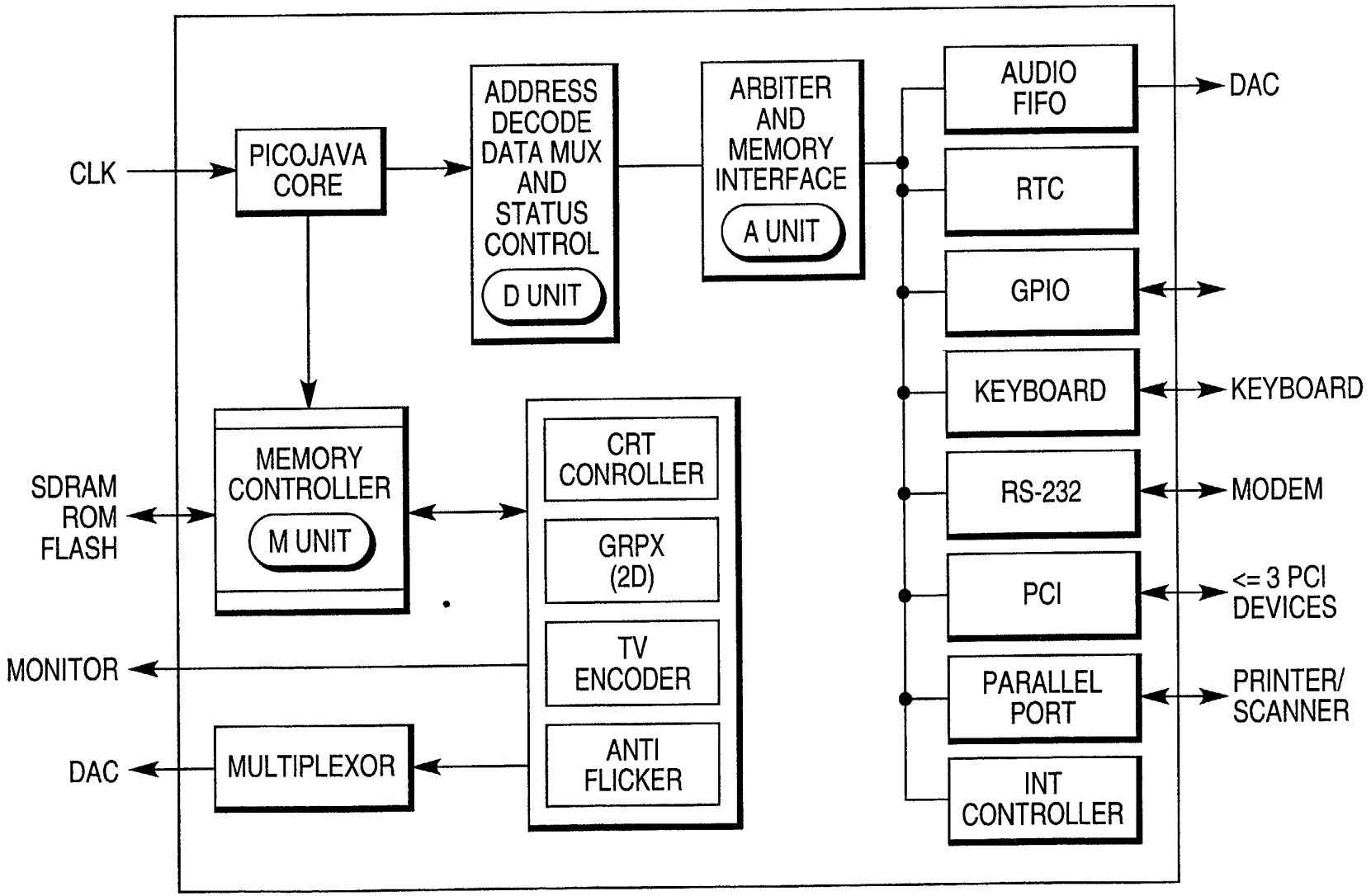


FIG. 22

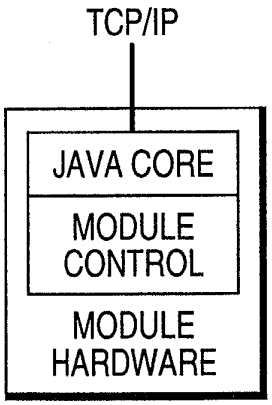


FIG. 23

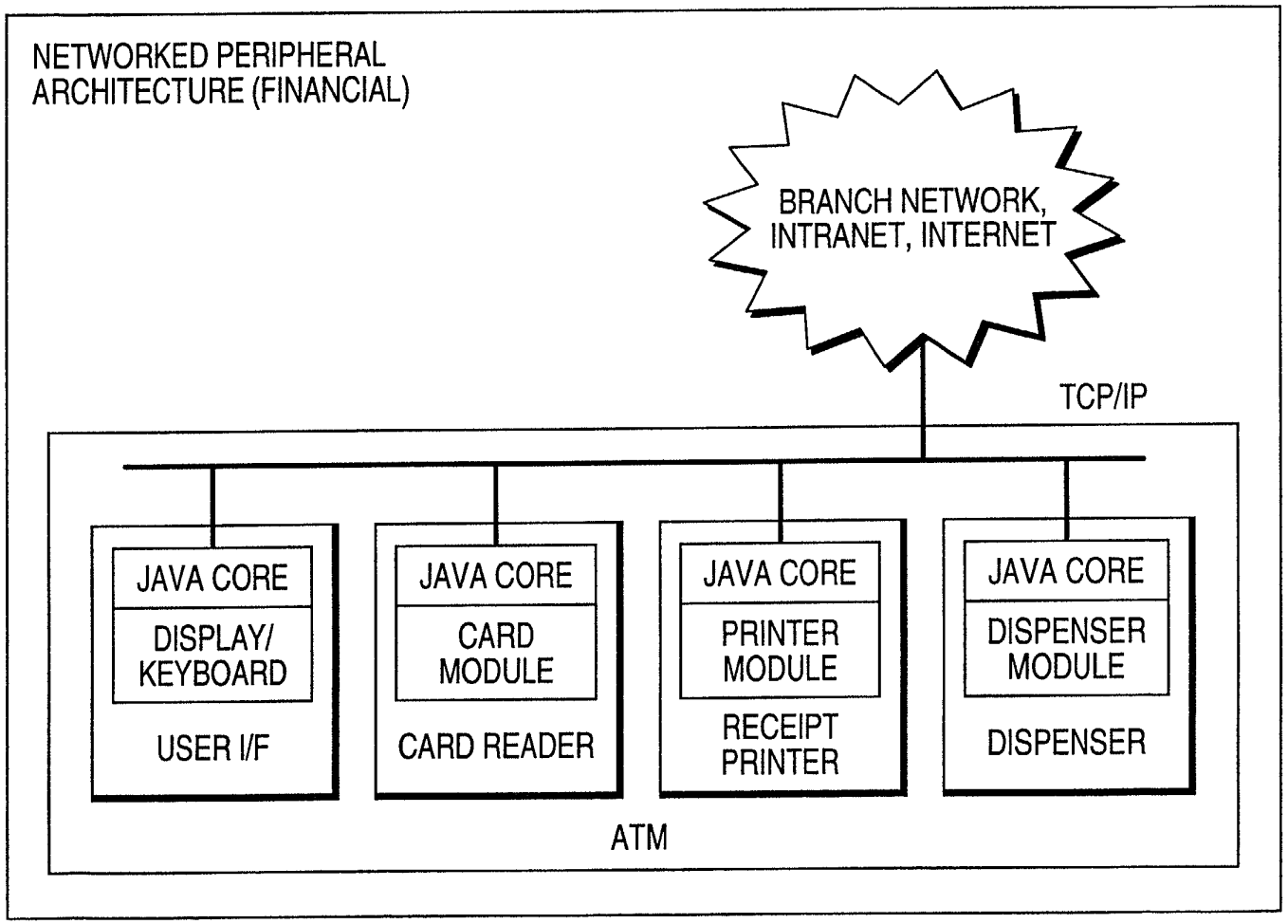


FIG. 24

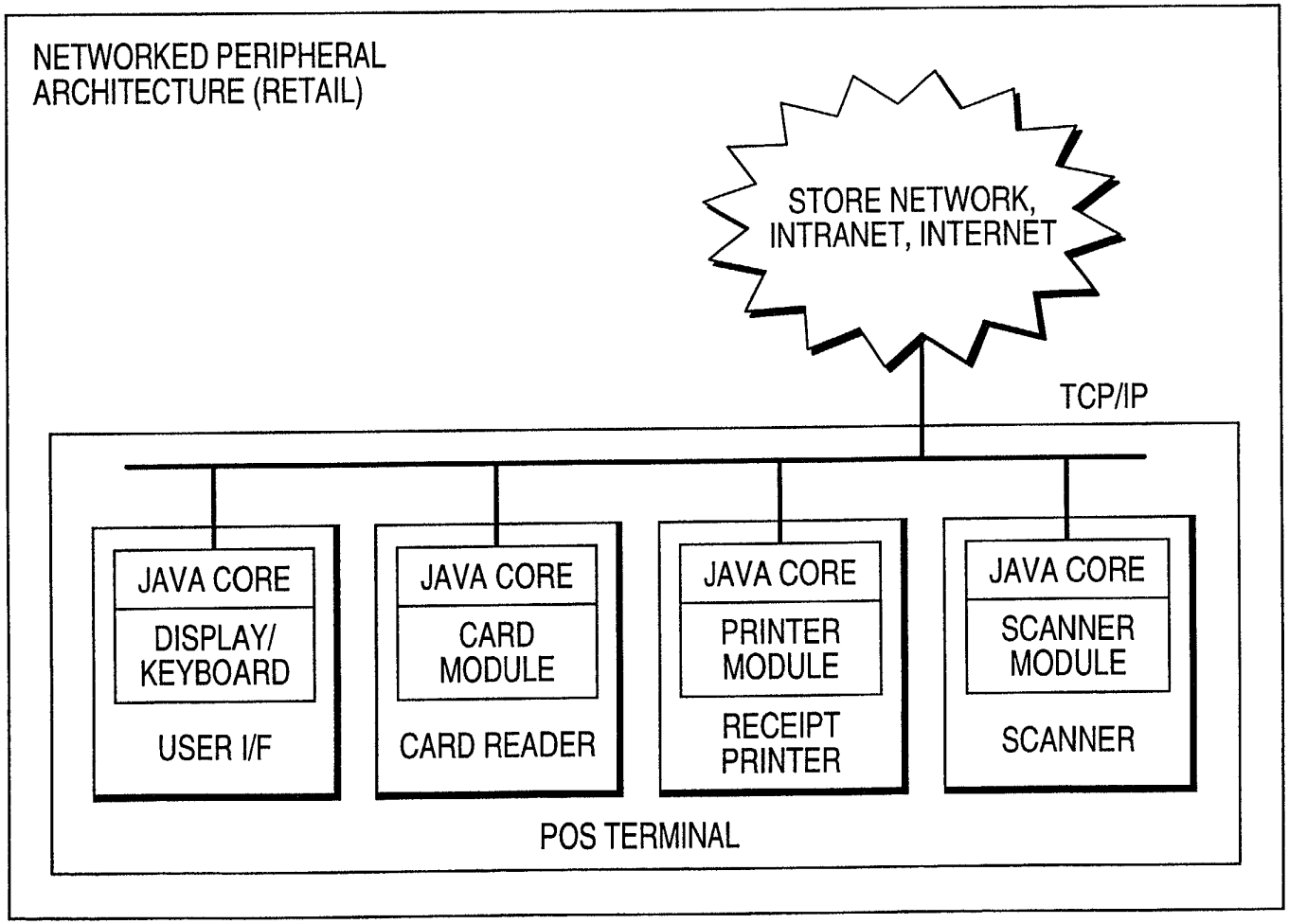


FIG. 25

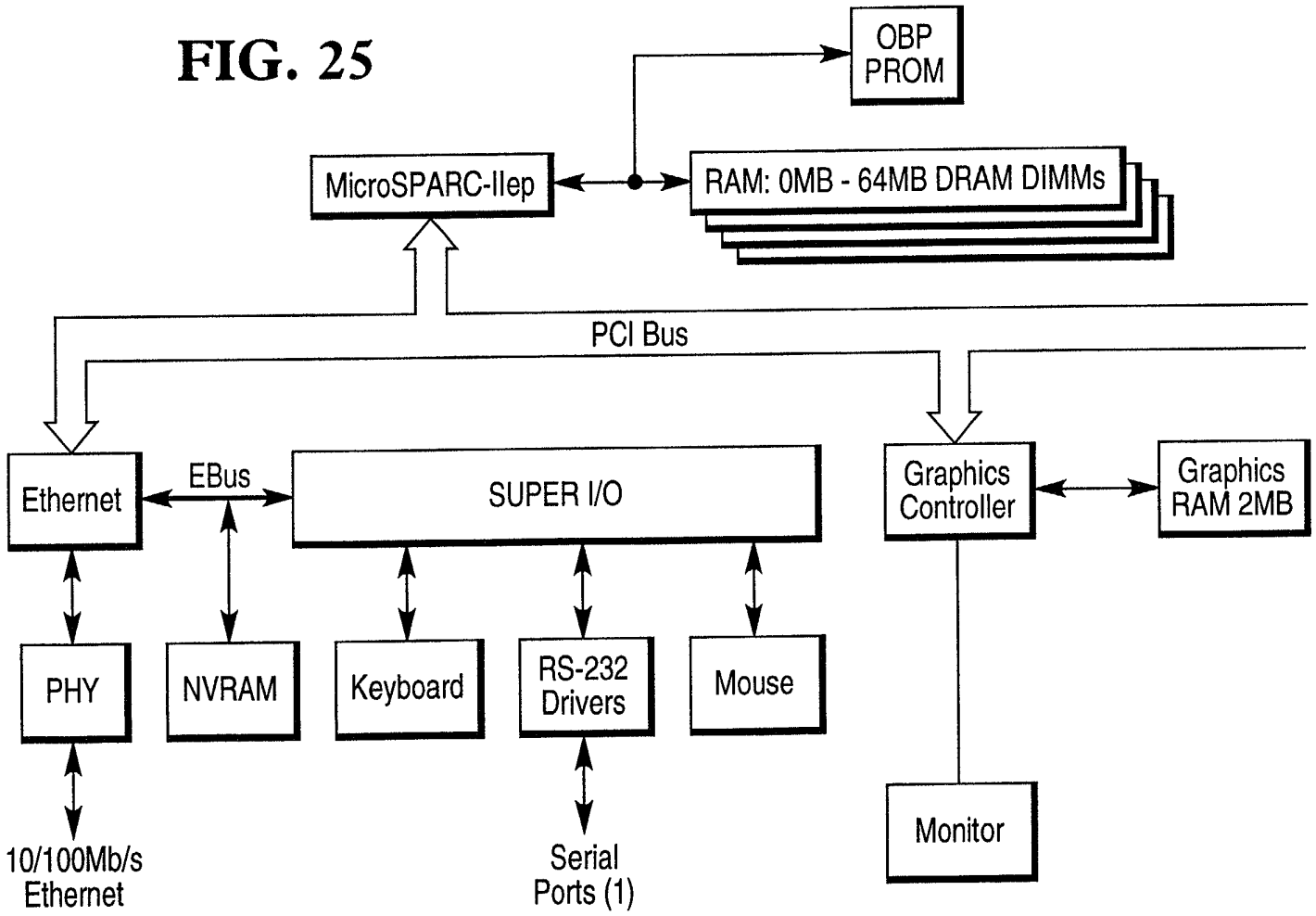


FIG. 26

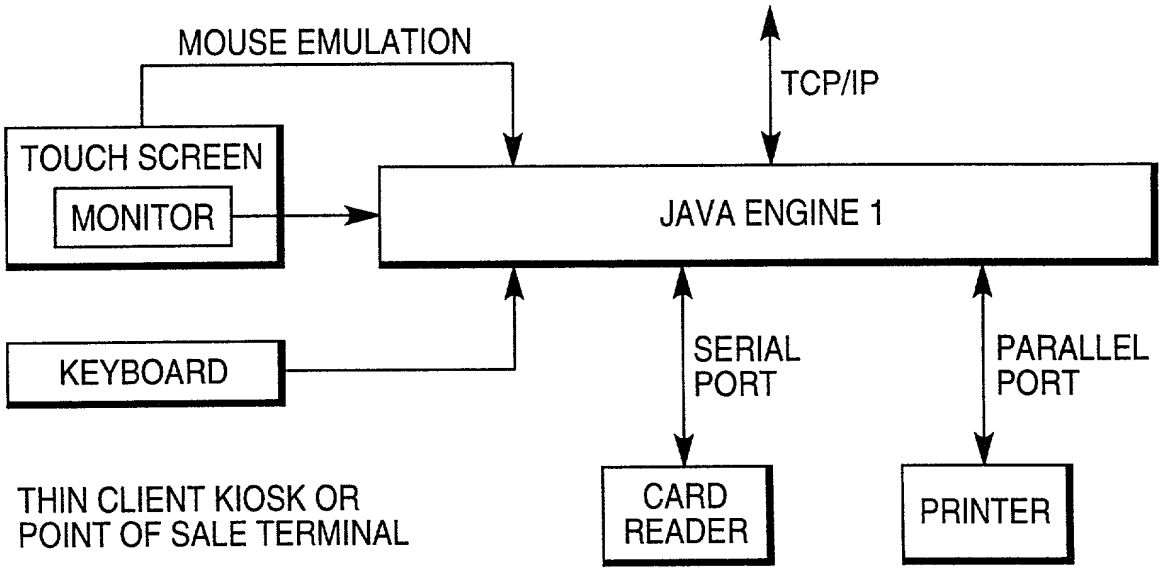

MOUSE EMULATION
TCP/IP
TOUCH SCREEN
MONITOR
JAVA ENGINE 1
KEYBOARD
SERIAL PORT
PARALLEL PORT
THIN CLIENT KIOSK OR POINT OF SALE TERMINAL
CARD READER
PRINTER

FIG. 27

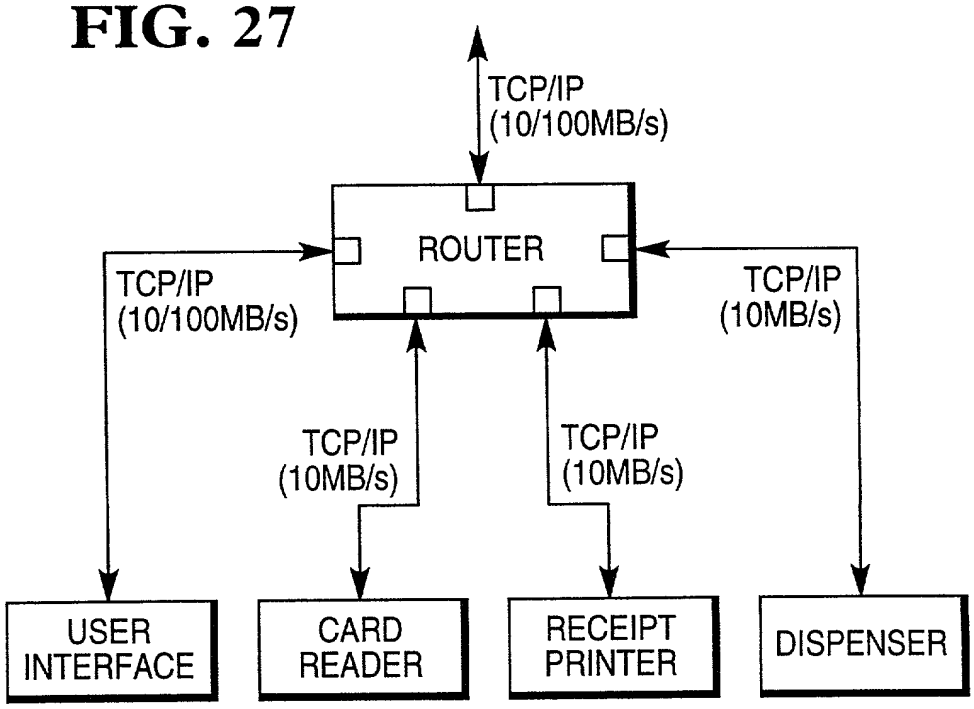

TCP/IP (10/100MB/s)
ROUTER
TCP/IP (10/100MB/s)
TCP/IP (10MB/s)
TCP/IP (10MB/s)
TCP/IP (10MB/s)
USER INTERFACE
CARD READER
RECEIPT PRINTER
DISPENSER

FIG. 28

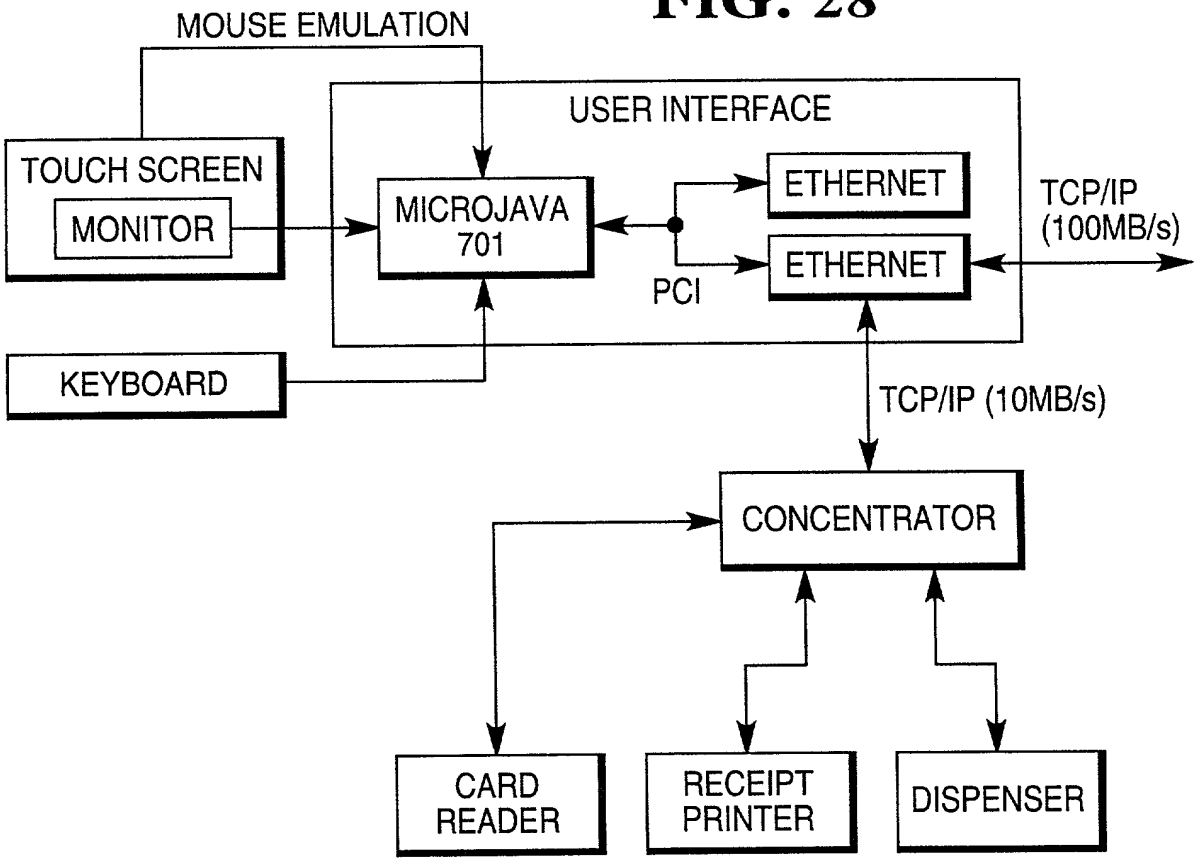
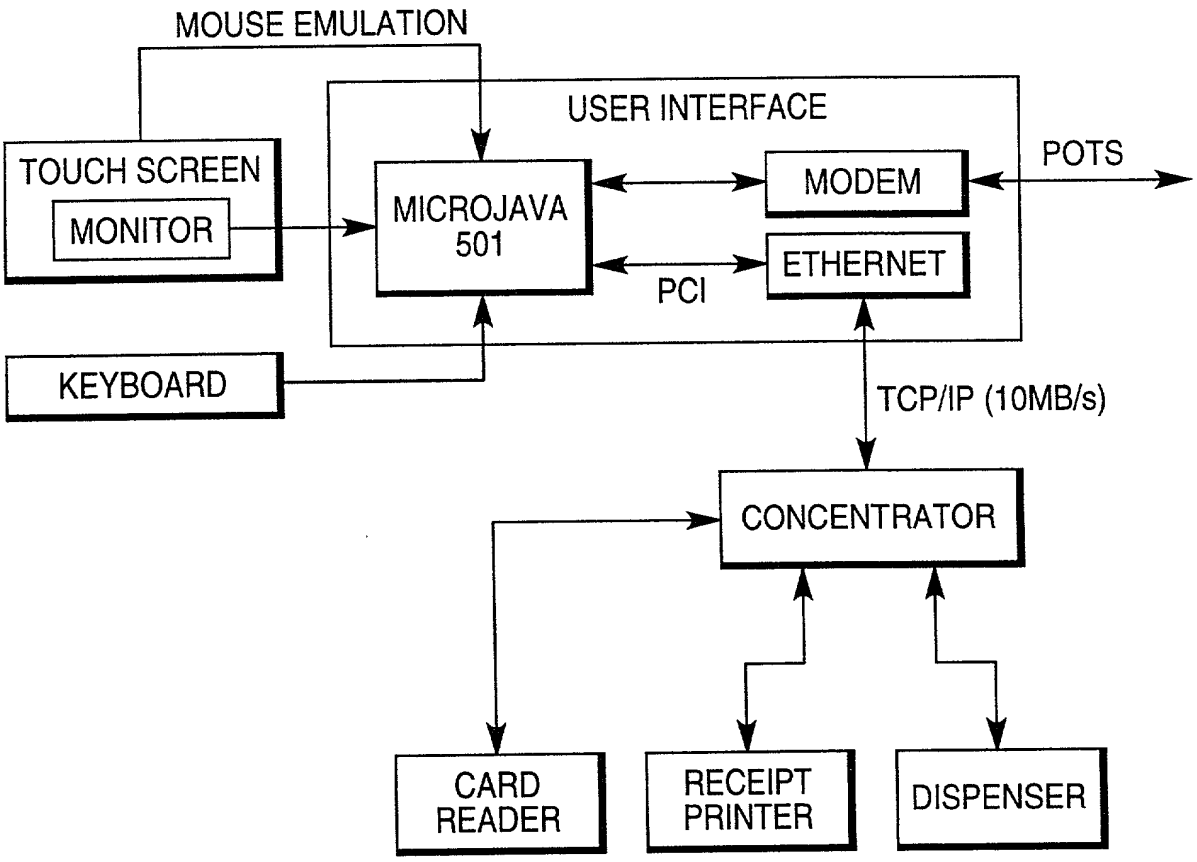

MOUSE EMULATION
USER INTERFACE
TOUCH SCREEN
MONITOR
MICROJAVA 701
ETHERNET
ETHERNET
TCP/IP (100MB/s)
PCI
KEYBOARD
TCP/IP (10MB/s)
CONCENTRATOR
CARD READER
RECEIPT PRINTER
DISPENSER

FIG. 29


MOUSE EMULATION
USER INTERFACE
TOUCH SCREEN
MONITOR
MICROJAVA 501
MODEM
POTS
ETHERNET
PCI
KEYBOARD
TCP/IP (10MB/s)
CONCENTRATOR
CARD READER
RECEIPT PRINTER
DISPENSER

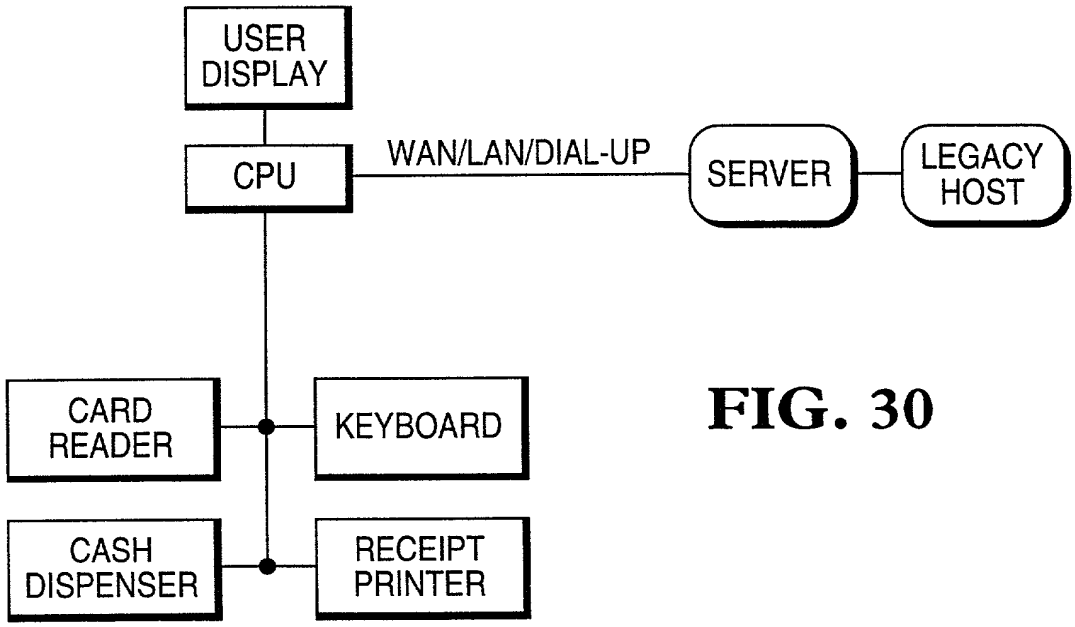

USER DISPLAY
CPU
WAN/LAN/DIAL-UP
SERVER
LEGACY HOST
CARD READER
KEYBOARD
CASH DISPENSER
RECEIPT PRINTER

FIG. 30

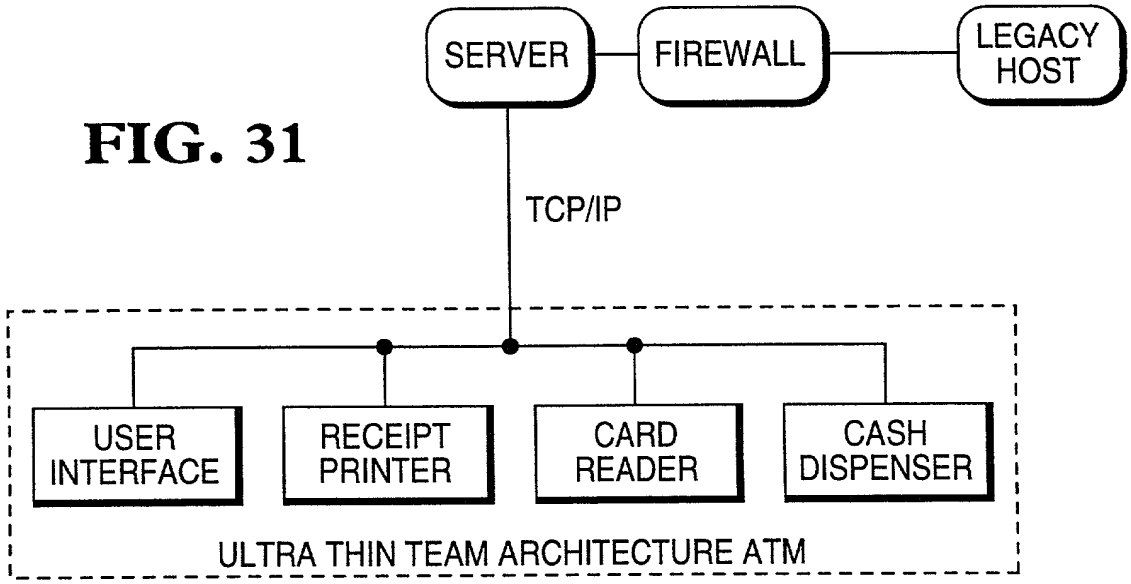
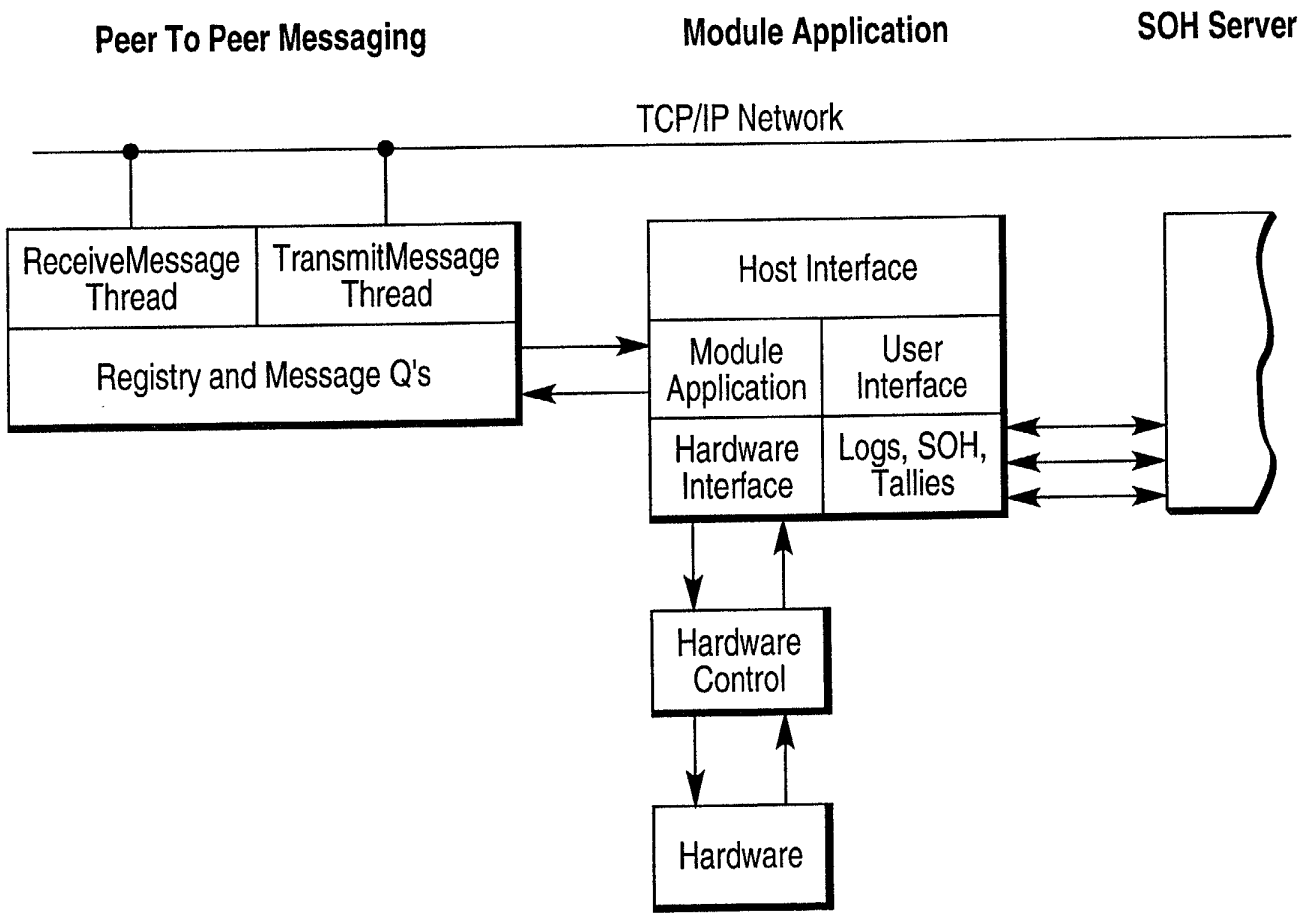

SERVER
FIREWALL
LEGACY HOST
TCP/IP
USER INTERFACE
RECEIPT PRINTER
CARD READER
CASH DISPENSER
ULTRA THIN TEAM ARCHITECTURE ATM

FIG. 31

FIG. 32


Peer To Peer Messaging
Module Application
SOH Server
TCP/IP Network
ReceiveMessage Thread
TransmitMessage Thread
Registry and Message Q's
Host Interface
Module Application
User Interface
Hardware Interface
Logs, SOH, Tallies
Hardware Control
Hardware

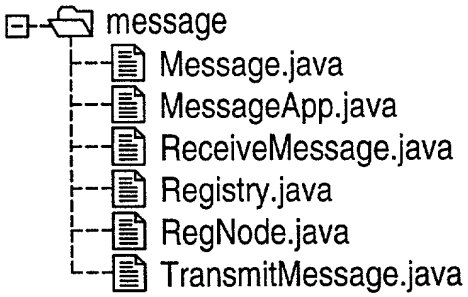


FIG. 33

FIG. 34

FIELD NAME	PURPOSE
message	A string object containing the actual message.
value	A value that can be associated with the above message.
me	The name of the sending module, for example, CARD_READER.
id	Unique ATM identification number given to every module in the team.
port	Port number of the server, when a return message can be sent.
scan	Identifies the message type.

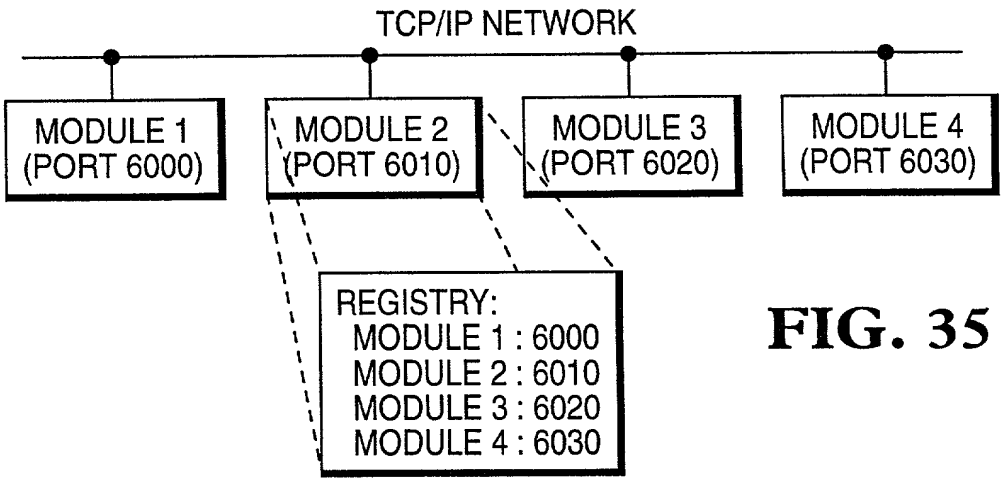


FIG. 35

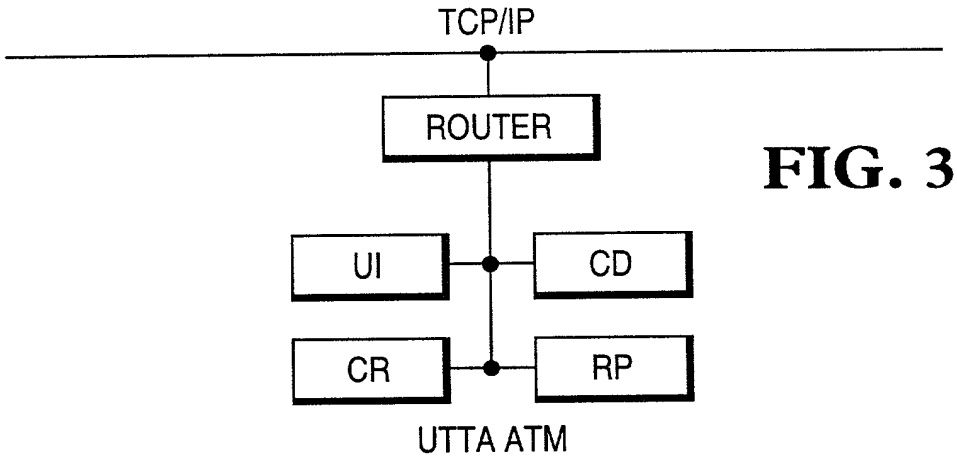
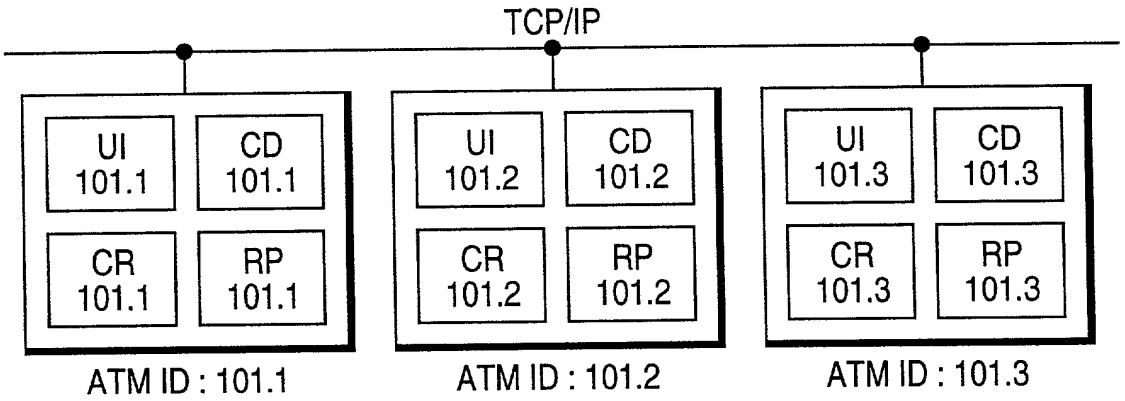


FIG. 36

FIG. 37



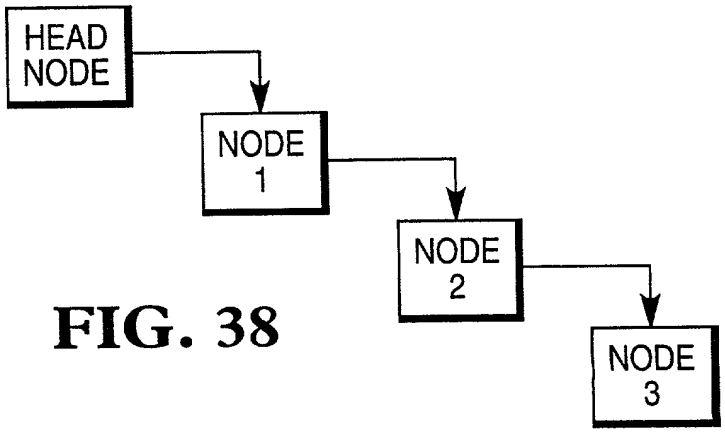


FIG. 38

FIG. 39

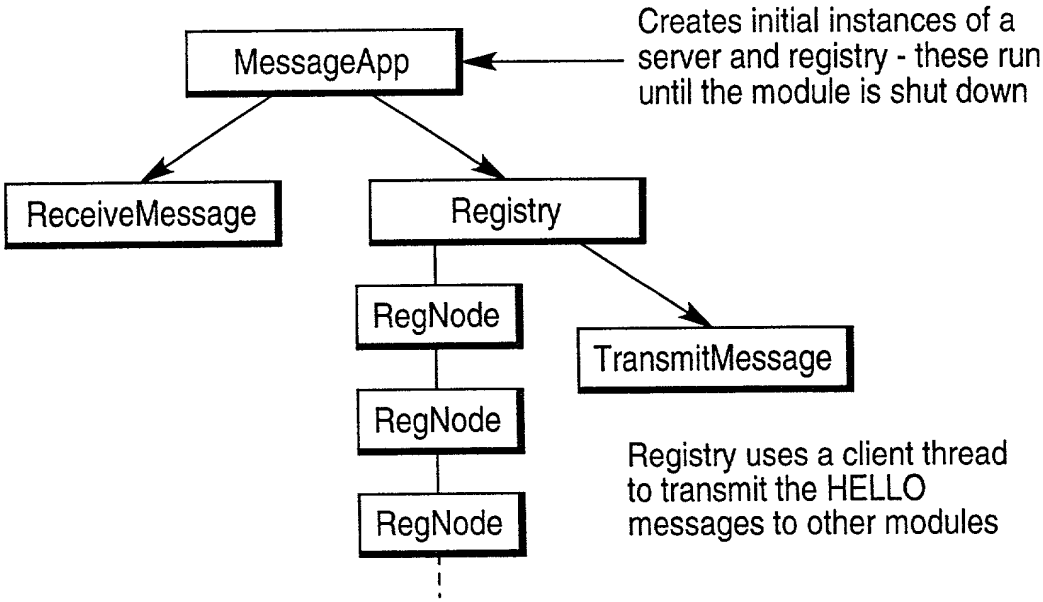
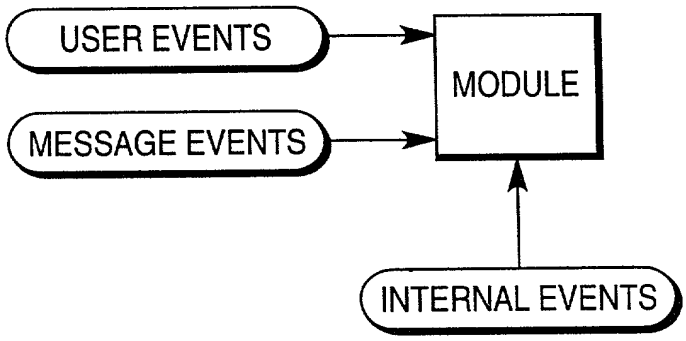

MessageApp
Creates initial instances of a server and registry - these run until the module is shut down
ReceiveMessage
Registry
RegNode
RegNode
RegNode
TransmitMessage
Registry uses a client thread to transmit the HELLO messages to other modules

FIG. 40


USER EVENTS
MESSAGE EVENTS
MODULE
INTERNAL EVENTS

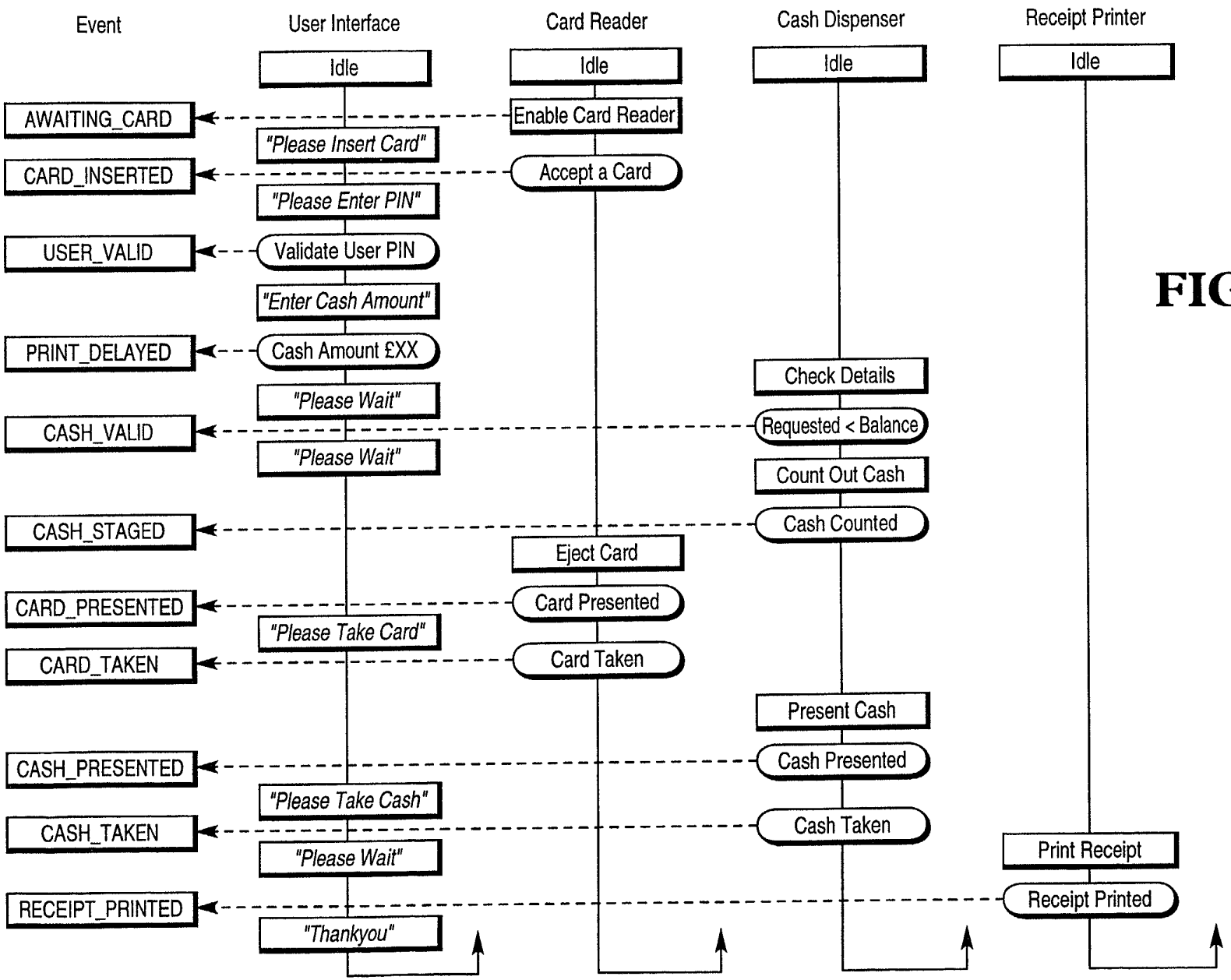


FIG. 41

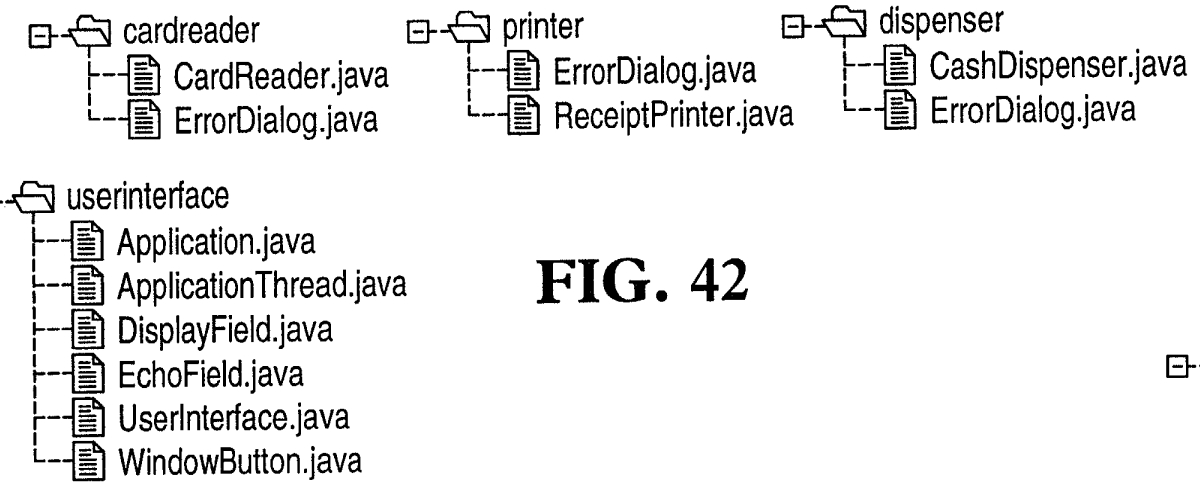


FIG. 42

FIG. 44

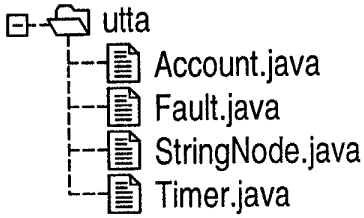


FIG. 45

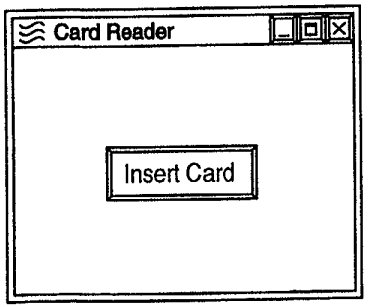
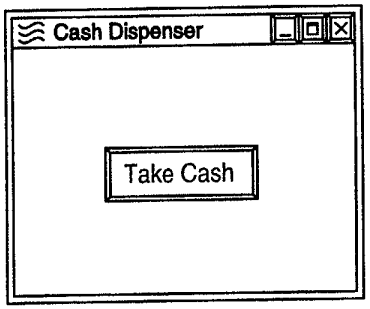


FIG. 46



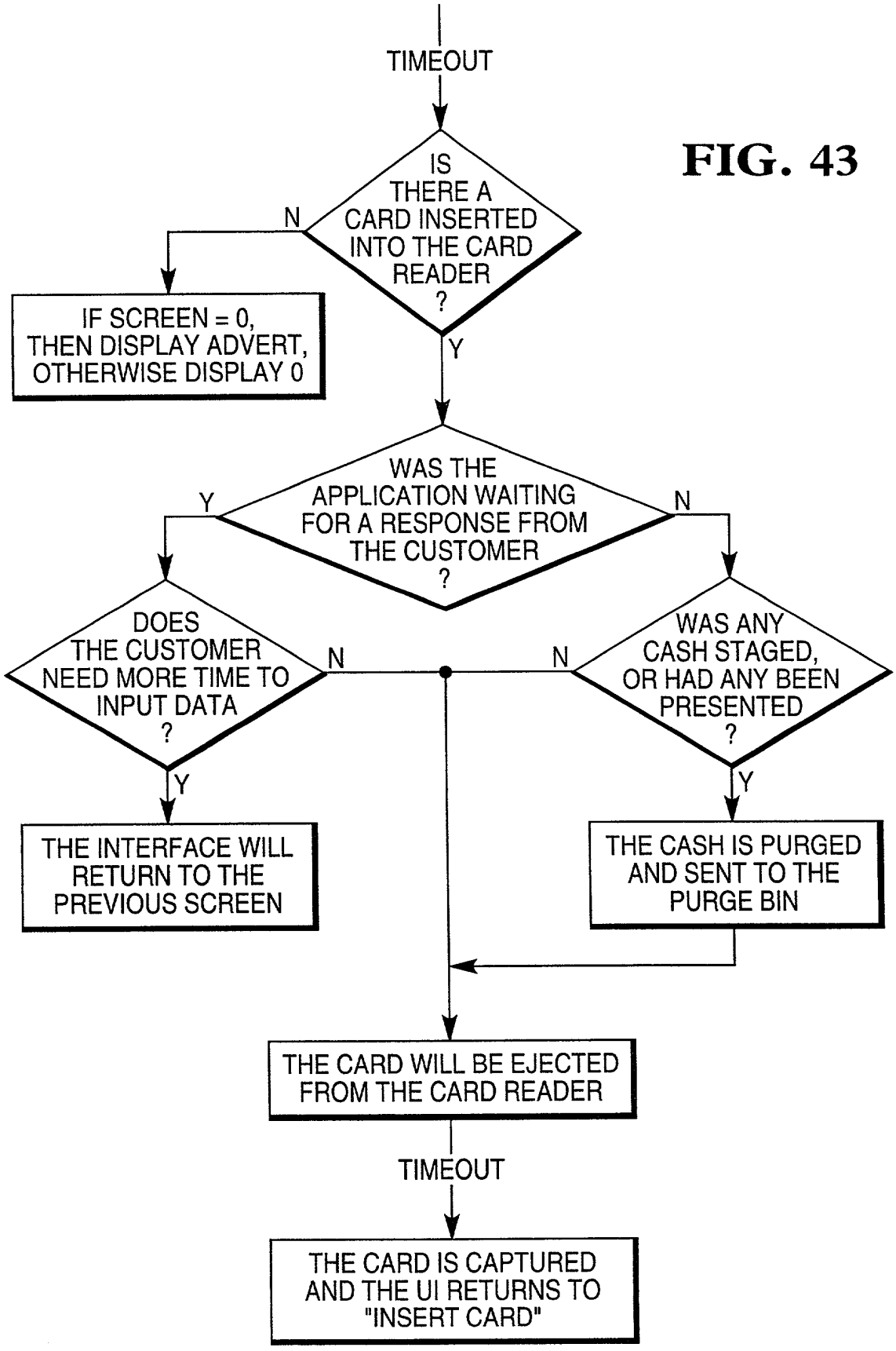


FIG. 43

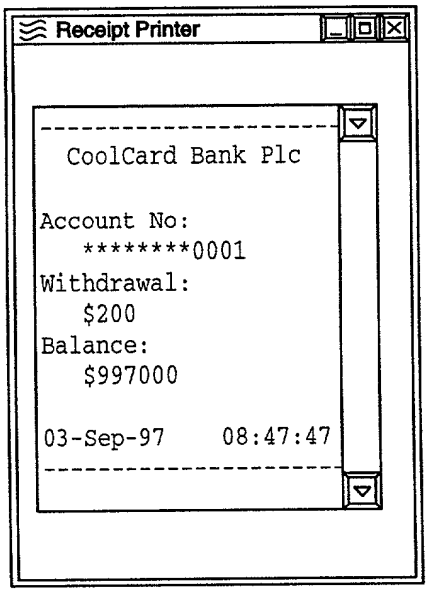

Receipt Printer
CoolCard Bank Plc
Account No:
********0001
Withdrawal:
$200
Balance:
$997000
03-Sep-97 08:47:47

FIG. 47

FIG. 48

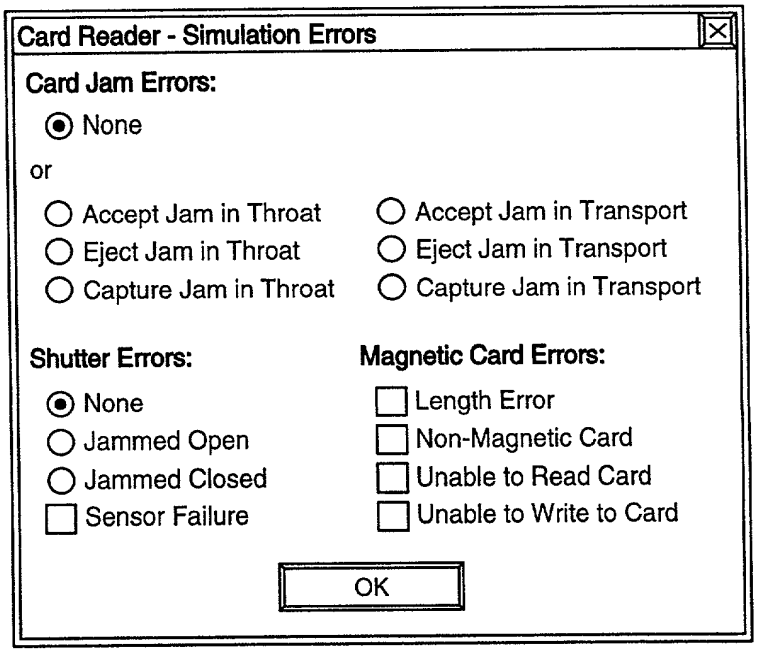

Card Reader - Simulation Errors
Card Jam Errors:
None
or
Accept Jam in Throat
Eject Jam in Throat
Capture Jam in Throat
Accept Jam in Transport
Eject Jam in Transport
Capture Jam in Transport
Shutter Errors:
None
Jammed Open
Jammed Closed
Sensor Failure
Magnetic Card Errors:
Length Error
Non-Magnetic Card
Unable to Read Card
Unable to Write to Card
OK

FIG. 49

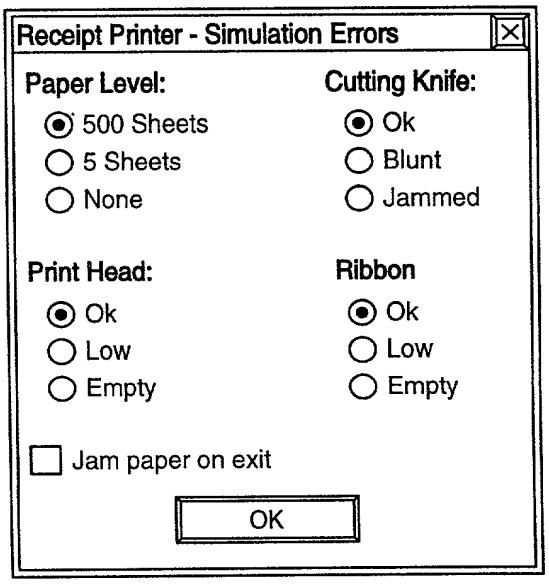
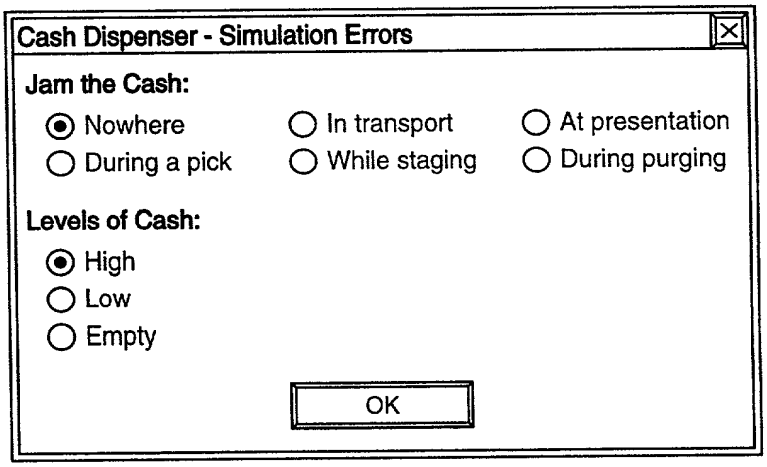

Receipt Printer - Simulation Errors
Paper Level:
500 Sheets
5 Sheets
None
Cutting Knife:
Ok
Blunt
Jammed
Print Head:
Ok
Low
Empty
Ribbon
Ok
Low
Empty
Jam paper on exit
OK

FIG. 50


Cash Dispenser - Simulation Errors
Jam the Cash:
Nowhere
During a pick
In transport
While staging
At presentation
During purging
Levels of Cash:
High
Low
Empty
OK

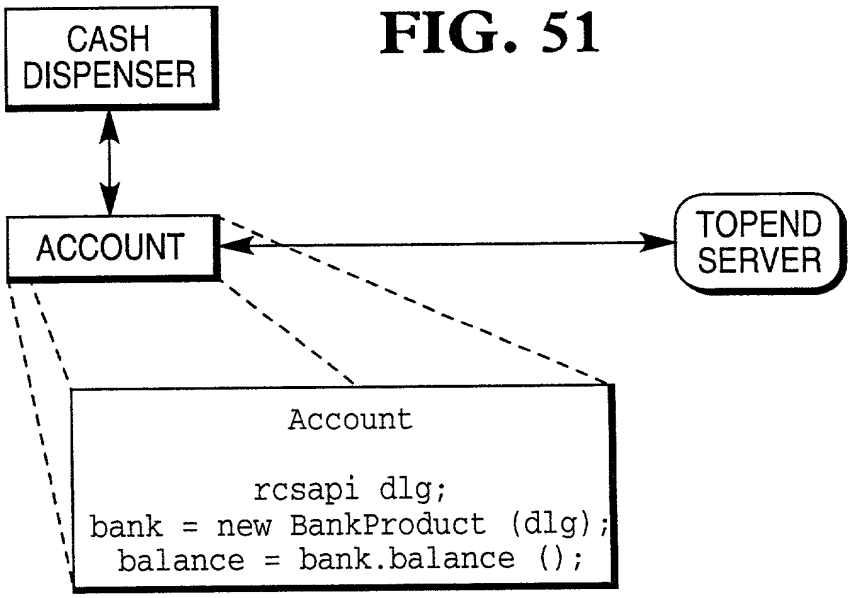


FIG. 51

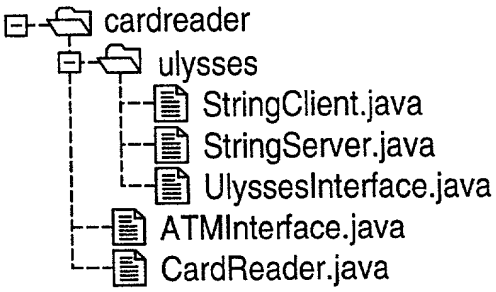


FIG. 52

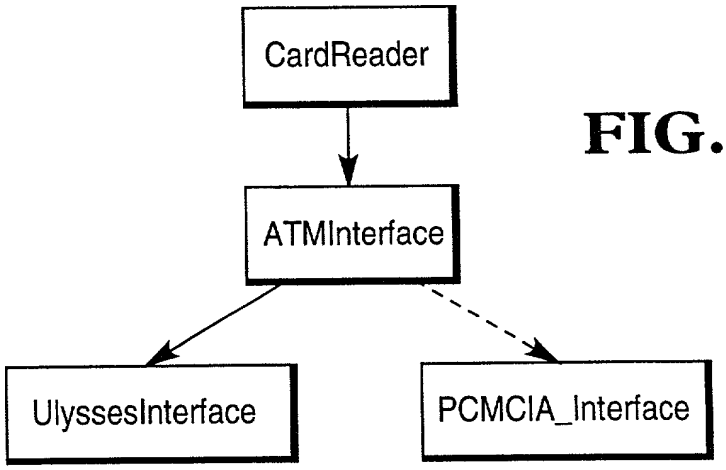


FIG. 53

FIG. 54

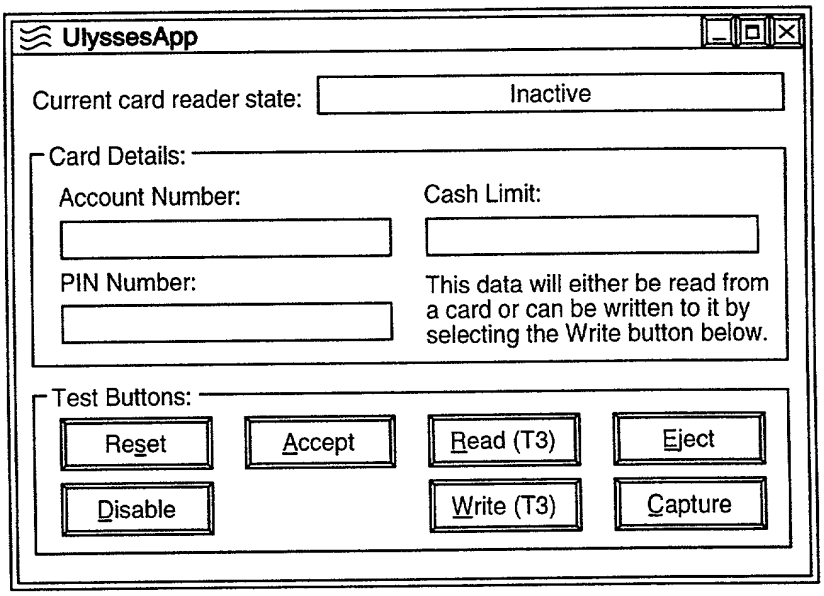

UlyssesApp
Current card reader state:
Inactive
Card Details:
Account Number:
Cash Limit:
PIN Number:
This data will either be read from a card or can be written to it by selecting the Write button below.
Test Buttons:
Reset
Accept
Read (T3)
Eject
Disable
Write (T3)
Capture

FIG. 55

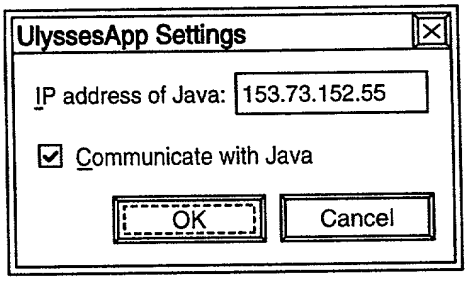

UlyssesApp Settings
IP address of Java:
153.73.152.55
Communicate with Java
OK
Cancel

FIG. 56A

JAVA CLASS FILES	SUMMARY
Account	All access to the TopEnd server was achieved through the use of this class. It defined a simple bank account, with a balance and the amount of cash involved in the last transaction. The UTTA modules could access bank details through this class without concern about contacting TopEnd.
Fault	Fault defined a whole set of constant variables which were used by the three modules that simulated hardware errors - namely, the card reader, receipt printer, and cash dispenser. Its use was primarily to make the code easier to read.
StringNode	Because Java did not allow for the use of templates, for each kind of linked-list created the nodes for different data types had to be redefined. StringNode were used to create linked-lists where each node contained a simple String instance.
Timer	Timer was used throughout the code for the purpose of delaying a module's application flow. Delays were used where the software was simulating what the hardware does - for example, one was used within the cash dispenser to halt execution while it 'pretended' to transport the cash from the cassette to the staging area.
ATMInterface	This class provided standard interface methods for the card reader (Ulysses version) to talk to hardware. The CardReader class only knew about ATMInterface, however, it was actually UlyssesInterface that is instantiated to do the work.
CardReader	The Java frame that contained application-control flow code for running a card reader module. There were two versions of this - the first which was only a software simulation, whereas the 'Hardware' version sent and received commands to control a real card reader.
ErrorDialog	The frame containing all the simulation error choices associated with the normal card reader. Note, that the ErrorDialog class was not found in the hardware version of the card reader as it was not used.

FIG. 56B

JAVA CLASS FILES	SUMMARY
StringClient	An extension of the Thread class that allowed for the sending of simple String based messages across a network using sockets. This class was used to send card reader commands to the Windows-Ulysses program.
StringServer	An extension of the Thread class that continuously listened on a socket for incoming String based messages.
UlyssesInterface	Deals with the command messages from the CardReader and also the response messages that were received from Windows (via StringServer). The class simply passed on the command messages, however, the incoming ones were deciphered and the correct response was passed back to the CardReader.
CashDispenser	The Java frame that contained application-control flow code for running a cash dispenser module
ErrorDialog	The frame containing all the simulation error choices associated with the above cash dispenser class.
Message	Message was a serialised object class, designed for sending across a network from module to module, being reconstructed at the receiving end. As well as the actual message, the Message class contained additional fields for information relating to the sending module.
MessageApp	This is the class that the four modules actually imported to handle their messages for them. It controlled both the ReceiveMessage thread and the registry of active modules. It contained interface methods that allowed a module to send and receive the messages.
ReceiveMessage	An extension of the Thread class that continuously listened on a socket for incoming messages from other modules. Once received, the class would pass the message back up to MessageApp for collection by a module.
Registry	Registry contains all the methods that dealt with accessing and maintaining a list of active modules in the ATM. The same instance of a Registy class within a Java virtual machine (such as the card reader) would be used and accessed by MessageApp, ReceiveMessage, and TransmitMessage.
RegNode	Defined the data that was stored at each location in a module's registry. It is used within a linked-list.
TransmitMessage	An extension of the Thread class that allowed for the sending of messages (based on the Message object) across a network.

FIG. 56C

JAVA CLASS FILES	SUMMARY
ReceiptPrinter	The Java frame that contained application-control flow code for running a receipt printer module
ErrorDialog	The frame containing all the simulation error choices associated with the above receipt printer class.
Application	This is the main application-control flow for the user interface. It is the most complex of the four control flows as the display is the main form of interaction with the user. As well as the control flow, the class also contained code required to draw the various screens within its frame.
ApplicationThread	The thread created within this class was used to control two things - the display of screens on the main user interface, and to monitor the limits set for the various time-outs. If one occured, then this class informed the main Application of the event.
DisplayField	DisplayField objects were used within the UserInterface class to provide text output to the screen. Using a larger font than normal, this class let the programmer position text accurately that will respond automatically to changes in the screen size and position.
EchoField	EchoField was used in several places within the main interface display. Firstly, during the PIN entry (or PIN entry retry) screen, an EchoField object was used to display '*' characters as each number was entered. It was also used to show user input in the cash entry screens, except this time no attempt to hide the characters was made.
UserInterface	The main frame for the user interface. This only created an area for the display to happen. All the control flow and display code could be found within the Application class.
WindowButton	WindowButton was a special implementation of a button, that allowed for large on-screen buttons (that could contain images), that can expand or contract in size according to the size of the user interface. These 'buttons' could either contain just text, a picture, or a picture without the ability to press the button.

FIG. 57

C++ FILE NAME	SUMMARY
StdAfx.h	This is part of the standard included files for any Visual C++ MFC application.
StdAfx.cpp	A Visual C++ MFC specific file included in the prototype code for other user-created code to function.
Resource.h	A Visual C++ specific file that contains the code for Windows, controls, icons, etc., used within the prototype.
UlyssesApp.h	Defines variables and functions used by the UlyssesApp class that was actually responsible for creating and running the main application.
UlyssesApp.cpp	Contains the function definitions for starting up the application and creating the dialog instance.
Message.h	Various user defined message maps were used within the main dialog class to receive messages from the Java virtual machine. These messages are defined in this file, where they can then be used by any class or function that includes it.
Server.h	To properly run the server listening for Java messages in its own thread, the code for it was declared in a separate file from the main dialog. Server.h contained two functions - one that actually listens for the text messages on the network, and another to post those messages back to the dialog window.
UlyssesDlg.h	Defines the variables and functions used by the main dialog class.
UlyssesDlg.cpp	This is the main application class file for the UlyssesApp prototype. This class was responsible for creating and setting up the Java server, registering itself with hardware (via the Ulysses API), and also passing messages between Java and the hardware.
SettingsDlg.h	Defines the variables and functions used by the settings dialog box that a user could pop up.
SettingsDlg.cpp	This contains the code for the settings box that allowed a user to decide if and where messages to Java are to be sent.

FIG. 58

Response to...	T-Code	Meaning
ACCEPT	0	Ok, reader enabled, no card present. UNSOLIC_RESPONSE to follow.
	1	Ok, card staged in reader, reader disabled. No UNSOLIC_RESPONSE to follow.
	2	Track not supported, read disabled. All commands available.
	3	Device inoperative - do not issue any commands until severity is checked.
	4	Card jam - eject or capture card.
	5	Invalid or jammed card in reader, card ejected and UNSOLIC_RESPONSE message to follow.
READ	0	Successful read.
	1	Read error, all commands available.
	2	Track not supported, all commands available.
	3	Device inoperative - do not issue any commands until severity is checked.
	4	Card jam - eject or capture card, or card removed, or too many read errors.
	5	Blank track, all commands available.
	6	No card present, all commands available.
WRITE	0	Card written Ok.
	1	Write error, all commands available.
	2	Track 3 write not supported, all other commands available.
	3	Device inoperative - do not issue any more commands until severity is checked.
	4	Card jam - eject or capture card, OR too many write errors.
	5	Write error, invalid data (length or format).
	6	No card present, all commands available.
EJECT	0	Card ejected, can be removed - all commands available.
	1	No card present, all commands available.
	2	Card captured, reader clear - all commands available.
	3	Device inoperative - do not issue any commands until severity is checked.
CAPTURE	0	Card captured Ok.
	1	Transport clear, no card to capture.
	2	Transport clear, card removed by customer
	3	Device inoperative - do not issue any commands until severity is checked.
UNSOLICITED	0	Card removed Ok.
	1	Card detected entering reader.
	2	Card inserted and staged Ok.
	3	Device inoperative - do not issue any commands until severity is checked.
	4	Card jam - eject or capture card.
	5	Invalid card inserted in reader. Card ejected, UNSOLIC to follow.
	6	Shutter jammed closed on attempted card entry.